SYBASE®

Programmer's Reference for COBOL

Mainframe Connect Client Option

12.6

IBM CICS, IMS, and MVS

DOCUMENT ID: DC36470-01-1260-01

LAST REVISED: March 2005

Copyright © 1991-2005 by Sybase, Inc. All rights reserved.

This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.

Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor. Upgrades are provided only at regularly scheduled software release dates. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase, the Sybase logo, ADA Workbench, Adaptable Windowing Environment, Adaptive Component Architecture, Adaptive Server, Adaptive Server Anywhere, Adaptive Server Enterprise, Adaptive Server Enterprise Monitor, Adaptive Server Enterprise Replication, Adaptive Server Everywhere, Adaptive Warehouse, Afaria, Answers Anywhere, Anywhere Studio, Application Manager, AppModeler, APT Workbench, APT-Build, APT-Edit, APT-Execute, APT-Translator, APT-Library, AvantGo Mobile Delivery, AvantGo Mobile Inspection, AvantGo Mobile Marketing Channel, AvantGo Mobile Pharma, AvantGo Mobile Sales, AvantGo Pylon, AvantGo Pylon Application Server, AvantGo Pylon Conduit, AvantGo Pylon PIM Server, AvantGo Pylon Pro, Backup Server, BizTracker, ClearConnect, Client-Library, Client Services, Convoy/DM, Copernicus, Data Pipeline, Data Workbench, DataArchitect, Database Analyzer, DataExpress, DataServer, DataWindow, DataWindow .NET, DB-Library, dbQueue, Developers Workbench, Direct Connect Anywhere, DirectConnect, Distribution Director, e-ADK, E-Anywhere, e-Biz Impact, e-Biz Integrator, E-Whatever, EC Gateway, ECMAP, ECRTP, eFulfillment Accelerator, Embedded SQL, EMS, Enterprise Application Studio, Enterprise Client/Server, Enterprise Connect, Enterprise Data Studio, Enterprise Manager, Enterprise SQL Server Manager, Enterprise Work Architecture, Enterprise Work Designer, Enterprise Work Modeler, eProcurement Accelerator, EWA, Financial Fusion, Financial Fusion Server, Gateway Manager, GlobalFIX, iAnywhere, iAnywhere Solutions, ImpactNow, Industry Warehouse Studio, InfoMaker, Information Anywhere, Information Everywhere, InformationConnect, InternetBuilder, iScript, Jaguar CTS, jConnect for JDBC, M2M Anywhere, Mail Anywhere Studio, MainframeConnect, Maintenance Express, Manage Anywhere Studio, M-Business Channel, M-Business Network, M-Business Server, MDI Access Server, MDI Database Gateway, media.splash, MetaWorks, mFolio, Mirror Activator, MySupport, Net-Gateway, Net-Library, New Era of Networks, ObjectConnect, ObjectCycle, OmniConnect, OmniSQL Access Module, OmniSQL Toolkit, Open Biz, Open Client, Open ClientConnect, Open Client/Server, Open Client/Server Interfaces, Open Gateway, Open Server, Open ServerConnect, Open Solutions, Optima++, PB-Gen, PC APT Execute, PC DB-Net, PC Net Library, PocketBuilder, Pocket PowerBuilder, Power++, power.stop, PowerAMC, PowerBuilder, PowerBuilder Foundation Class Library, PowerDesigner, PowerDimensions, PowerDynamo, PowerScript, PowerSite, PowerSocket, Powersoft, PowerStage, PowerStudio, PowerTips, Powersoft Portfolio, Powersoft Professional, PowerWare Desktop, PowerWare Enterprise, ProcessAnalyst, QAnywhere, Rapport, RemoteWare, RepConnector, Replication Agent, Replication Driver, Replication Server, Replication Server Manager, Replication Toolkit, Report-Execute, Report Workbench, Resource Manager, RFID Anywhere, RW-DisplayLib, RW-Library, S-Designor, SDF, Secure SQL Server, Secure SQL Toolset, Security Guardian, SKILS, smart.partners, smart.parts, smart.script, SQL Advantage, SQL Anywhere, SQL Anywhere Studio, SQL Code Checker, SQL Debug, SQL Edit, SQL Edit/TPU, SQL Everywhere, SQL Modeler, SQL Remote, SQL Server, SQL Server Manager, SQL SMART, SQL Toolset, SQL Server/CFT, SQL Server/DBM, SQL Server SNMP SubAgent, SQL Station, SQLJ, STEP, SupportNow, S.W.I.F.T. Message Format Libraries, Sybase Central, Sybase Client/Server Interfaces, Sybase Financial Server, Sybase Gateways, Sybase IQ, Sybase MPP, Sybase SQL Desktop, Sybase SQL Lifecycle, Sybase SQL Workgroup, Sybase User Workbench, SybaseWare, Syber Financial, SyberAssist, SybFlex, SyBooks, System 10, System 11, System XI (logo), SystemTools, Tabular Data Stream, TradeForce, Transact-SQL, Translation Toolkit, UltraLite, UltraLite.NET, UNIBOM, Unilib, Uninull, Unisep, Unistring, URK Runtime Kit for UniCode, VisualWriter, VQL, WarehouseArchitect, Warehouse Control Center, Warehouse Studio, Warehouse WORKS, Watcom, Watcom SQL, Watcom SQL Server, Web Deployment Kit, Web.PB, Web.SQL, WebSights, WebViewer, WorkGroup SQL Server, XA-Library, XA-Server, XcelleNet, and XP Server are trademarks of Sybase, Inc. 11/04

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

All other company and product names used herein may be trademarks or registered trademarks of their respective companies.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

About This Book ..... **vii**

CHAPTER 1 **Open ClientConnect Processing** **1**
- What is Open ClientConnect? 1
- Understanding three- and two-tier environments 2
- Open ClientConnect communications 3
  - Three-tier (gateway-enabled) environment 4
  - Two-tier (gateway-less) environment 7
  - Communication flow 8
  - Open ClientConnect security 9
  - How to choose a network driver 9
- Compatibility 14
- Open ClientConnect Client-Library functions 14
- Using Client-Library functions 15
  - Basic control structures 15
- Steps in a simple program 18
- A simple language program 19
  - Setting up the Client-Library programming environment 19
  - Connecting to a server 19
  - Sending a command to the server 20
  - Processing the results of the command 20
  - Finishing up 21

CHAPTER 2 **Functions** **23**
- CTBBIND 25
- CTBCANCEL 35
- CTBCLOSE 37
- CTBCMDALLOC 41
- CTBCMDDROP 44
- CTBCMDPROPS 47
- CTBCOMMAND 51
- CTBCONALLOC 55
- CTBCONDROP 61

CTBCONFIG 64
CTBCONNECT 67
CTBCONPROPS 70
CTBDESCRIBE 78
CTBDIAG 86
CTBEXIT 104
CTBFETCH 107
CTBGETFORMAT 112
CTBINIT 117
CTBPARAM 120
CTBREMOTEPWD 127
CTBRESINFO 132
CTBRESULTS 137
CTBSEND 142
CSBCONFIG 148
CSBCONVERT 153
CSBCTXALLOC 160
CSBCTXDROP 162

CHAPTER 3 Topics 165
Buffers 165
CLIENTMSG structure 168
Customization 169
DATAFMT structure 170
Datatypes 174
Open ClientConnect datatypes 175
Error and message handling 177
The CS-EXTRA-INF property 179
Nulls 180
Properties 181
About the properties 185
Remote procedure calls (RPCs) 190
Stored procedure return parameters 191
Results 193
SERVERMSG structure 194
SQLCA structure 197
SQLCODE structure 198
Handles 199
Types of handles 199

APPENDIX A **Sample Language Requests 203**
Sample program - SYCTSAA5 203
SYCTSAP5 - sample language request 244

SYCTSAT5 - sample language request 286

APPENDIX B **Sample RPC Application 329**
Sample program – SYCTSAR5 330

APPENDIX C **Sybase Product Documentation 373**
Publications by content 373
Publications by audience 376

Index 379

About This Book

The Mainframe Connect Client Option *Programmer's Reference for COBOL* contains reference information for the COBOL version of Open ClientConnect™ Client-Library.

Note The Open ClientConnect Client-Library is a subset of the generic Sybase® Client-Library™.

This chapter contains the following topics:

- Audience
- Product name changes
- How to use this book
- Related documents
- Other sources of information
- Sybase certifications on the Web
- Sybase EBFs and software maintenance
- Conventions
- If you need help

Audience

The Mainframe Connect Client Option *Programmer's Reference for COBOL* is a reference manual for mainframe programmers who write client applications in the COBOL programming language using Open ClientConnect Client-Library.

This manual assumes that the programmer is familiar with the COBOL programming language and knows how to write programs under either CICS or IMS TM. This book does not contain instructions for writing COBOL programs. Rather, it describes the functions that can be called within your COBOL programs to enable them to communicate with remote servers.

Product name changes

The following table describes new names for products in the 12.6 release of the Mainframe Connect Integrated Product Set.

Old product names	New product name
• Open ClientConnect for CICS • Open ClientCONNECT for CICS	Mainframe Connect Client Option for CICS
• Open Client Connect for IMS and MVS • Open ClientCONNECT for IMS and MVS	Mainframe Connect Client Option for IMS and MVS
• Open ServerConnect® for CICS • Open ServerCONNECT for CICS	Mainframe Connect Server Option for CICS
• Open ServerConnect for IMS and MVS • Open ServerCONNECT for IMS and MVS	Mainframe Connect Server Option for IMS and MVS
• MainframeConnect™ for DB2 UDB • MainframeCONNECT for DB2/MVS-CICS	Mainframe Connect DB2 UDB Option for CICS
• DirectConnect for OS/390 • DirectCONNECT for DB2/MVS	DirectConnect for z/OS

The old product names are used throughout this book, except for on the title page.

Note This book also uses the terms MVS and OS/390 where the newer term z/OS would otherwise be used.

How to use this book

Table 1 shows where to find the information you need in this book.

Table 1: Chapter descriptions

Chapter	Contents
Chapter 1, “Open ClientConnect Processing”	An overview of Open ServerConnect including discussion of different kinds of client requests and explanations of how Open ServerConnect programs process them. **Note** Everyone who writes programs using Open ServerConnect should read this chapter.

Chapter	Contents
Chapter 3, “Topics”	Descriptions of Gateway-Library concepts, and information on how to accomplish specific programming tasks. This chapter discusses tasks, resources, and other topics that the application programmer needs to understand to write Gateway-Library applications. It includes a detailed discussion of the Gateway-Library cursor, dynamic SQL and Japanese language support, and a list of supported datatypes and models for structures used to store data.
Chapter 2, “Functions”	Reference pages for each Gateway-Library function. Each function description contains sections on functionality, syntax, explanatory comments and related functions, as well as an example.
Appendix A, “Sample Language Requests”	A sample COBOL application program that processes client language requests under CICS.
Appendix B, “Sample RPC Application”	A sample COBOL application program that processes client RPC requests under CICS.
Appendix C, “Sybase Product Documentation”	A list and description of all relevant Sybase product documentation.

Related documents

For a complete list and description of all related documents, see Appendix C, “Sybase Product Documentation.”

Other sources of information

Use the Sybase® Getting Started CD, the Sybase Technical Library CD, and the Technical Library Product Manuals Web site to learn more about your product:

- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the Technical Library CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader (downloadable at no charge from the Adobe Web site, using a link provided on the CD).
- The Technical Library CD contains product manuals and is included with your software. The DynaText reader (included on the Technical Library CD) allows you to access technical information about your product in an easy-to-use format.

  Refer to the *Technical Library Installation Guide* in your documentation package for instructions on installing and starting the Technical Library.

- The Technical Library Product Manuals Web site is an HTML version of the Technical Library CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

  To access the Technical Library Product Manuals Web site, go to Product Manuals at http://www.sybase.com/support/manuals/.

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ **Finding the latest information on product certifications**

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Select Products from the navigation bar on the left.
3. Select a product name from the product list and click Go.
4. Select the Certification Report filter, specify a time frame, and click Go.
5. Click a Certification Report title to display the report.

❖ **Creating a personalized view of the Sybase Web site (including support pages)**

Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

❖ **Finding the latest information on EBFs and software maintenance**

1. Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.
2. Select EBFs/Maintenance. Enter user name and password information, if prompted (for existing Web accounts) or create a new account (a free service).
3. Select a product.
4. Specify a time frame and click Go.

5 Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

Conventions

This section describes the syntax and style conventions used in this book.

Note Throughout this book, all references to Adaptive Server® Enterprise also apply to its predecessor, SQL Server. Also, Adaptive Server Enterprise (ASE) and Adaptive Server (AS) are used interchangeably.

Open ClientConnect uses eight-character function names, while other versions of Client-Library use longer names. This book uses the long version of Client-Library names with one exception: the eight-character version is used in syntax statements. For example, CTBCMDPROPS has eleven letters. In the syntax statement, it is written CTBCMDPR, using eight characters. You can use either version in your code.

Table 2 on page xi explains syntax conventions used in this book.

Table 2: Syntax conventions

Symbol	Explanation
( )	Parentheses indicate that parentheses are included as part of the command.
{ }	Braces indicate that you must choose at least one of the enclosed options. Do not type the braces when you type the option.
[ ]	Brackets indicate that you can choose one or more of the enclosed options, or none. Do not type the brackets when you type the options.
|	The vertical bar indicates that you can select only one of the options shown. Do not type the bar in your command.
,	The comma indicates that you can choose one or more of the options shown. Separate each choice by using a comma as part of the command.

Table 3 explains style conventions used in this book.

Table 3: Style conventions

This type of information	Looks like this
Gateway-Library function names	TDINIT, TDRESULT
Client-Library function names	CTBINIT, CTBRESULTS
Other executables (DB-Library routines, SQL commands) in text	the dbrpcparam routine, a select statement
Directory names, path names, and file names	*/usr/bin directory, interfaces* file
Variables	*n* bytes
Adaptive Server datatypes	datetime, float

This type of information	Looks like this
Sample code	01 BUFFER PIC S9(9) COMP SYNC. 01 BUFFER PIC X(n).
User input	01 BUFFER PIC X(n)
Client-Library and Gateway-Library function argument names	*BUFFER*, *RETCODE*
Client-Library function arguments that are input (I) or output (O)	*COMMAND* – (I) *RETCODE* – (O)
Names of objects stored on the mainframe	SYCTSAA5
Symbolic values used with function arguments, properties, and structure fields	CS-UNUSED, FMT-NAME, CS-SV-FATAL
Client-Library property names	CS-PASSWORD, CS-USERNAME
Client-Library and Gateway-Library datatypes	CS-CHAR, TDSCHAR

All other names and terms appear in this typeface.

If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

CHAPTER 1

Open ClientConnect Processing

This chapter includes the following topics:

- What is Open ClientConnect?
- Understanding three- and two-tier environments
- Open ClientConnect communications
- Open ClientConnect security
- How to choose a network driver
- Compatibility
- Open ClientConnect Client-Library functions
- Using Client-Library functions
- Steps in a simple program
- A simple language program

What is Open ClientConnect?

Open ClientConnect is a programming environment that provides Open Client Client-Library routines for use in building mainframe client applications.

Open ClientConnect runs on an IBM System/390 or plug-compatible mainframe computer. It uses the LU 6.2 or TCP/IP communications protocols and is available for CICS, IMS TM, and native MVS host transaction processors.

Note Some information in this guide is specific to version 4.0 of Open ClientConnect. This information will apply to Open ClientConnect for CICS only. All other information will apply both to Open ClientConnect for CICS and Open ClientConnect for IMS and MVS.

Open ClientConnect applications can communicate with two kinds of servers:

- Adaptive Server Enterprise and Open Server™ on PCs and several mid-range UNIX platforms.
- Open ServerConnect applications running in a separate region on the mainframe.

Open ClientConnect applications can send requests to Adaptive Server Enterprise, Open ServerConnect applications, and MainframeConnect™ for DB2 UDB (or OmniSQL Access Module™ for DB2).

Adaptive Server Enterprise

Open ClientConnect applications can send requests to Adaptive Server Enterprise indirectly through either of the following:

- The three-tier (gateway-enabled) environment using Mainframe ClientConnect (MCC).
- The two-tier (gateway-less) environment using TCP. See “Two-tier (gateway-less) environment” on page 7 for more information on two-tier environments.

Open ServerConnect

Open ClientConnect applications can send requests directly to Open ServerConnect running in a different CICS region. If using TCP, Open ClientConnect may send requests to Open ServerConnect running in the same CICS region.

Note Due to an IBM SNA restriction, connections from Open ClientConnect to Open ServerConnect require that they reside in different regions when connecting through LU 6.2. For TCP/IP, they can reside in the same region.

Understanding three- and two-tier environments

Open ClientConnect supports both three-tier (gateway-enabled) and two-tier (gateway-less) environments. When installing and using Open ClientConnect, follow the instructions in this book that are specific to your environment.

Three-tier (gateway-enabled) environments

If you use SNA as your protocol, you must use a three-tier environment for routing.

Note The DirectConnect product no longer comes with an MCC for TCP. A three-tier (gateway-enabled) environment using TCP as your protocol is no longer an option.

Two-tier (gateway-less)

If you have standardized to TCP for connectivity from CICS to Adaptive Server Enterprise, you must use the two-tier environment for routing. The two-tier environment allows Open ClientConnect to directly login to an Adaptive Server Enterprise, which eliminates the need for an MCC gateway.

An Open ClientConnect network configuration using two-tier (gateway-less) processing consists of the following:

- A host-based client, which is an Open ClientConnect program running under CICS. The client program selects a server and sends requests to that server.
- A server, which can be any server that Open ClientConnect applications can access, including servers on the LAN—Sybase Open Servers and Adaptive Server Enterprises—as well as Open ServerConnect running in a separate CICS region. If you use TCP as your protocol, Open ClientConnect may access servers running in the same CICS region.

Open ClientConnect communications

This section describes Open ClientConnect communications in three-tier and two-tier environments. It also explains the communication flow for both environments.

Note Although it is not shown in any of the following figures, Open ClientConnect for CICS also works with Open ServerConnect for IMS and MVS.

Three-tier (gateway-enabled) environment

The following figures show a basic Open ClientConnect configuration for CICS in three-tier (gateway-enabled) SNA and TCP/IP environments:

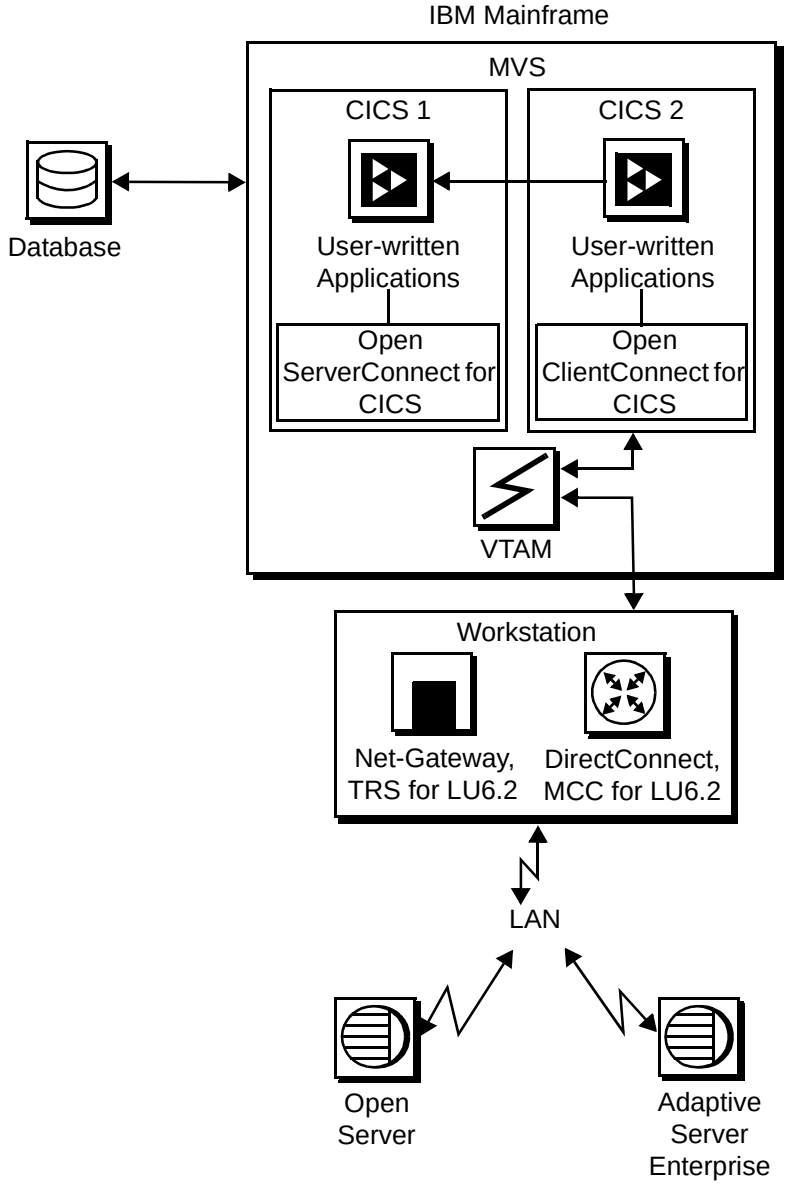
- Figure 1-1 on page 5
- Figure 1-2 on page 6

Note For three-tier, gateway-enabled environments, DirectConnect 11.1 (or Net-Gateway™ version 3.0.1 or higher) is a required companion product for full-feature compatibility with Open ServerConnect version 4.0 and Open ClientConnect version 3.2.

Three-tier SNA environment

Figure 1-1 shows Open ClientConnect communication in a three-tier (gateway-enabled) SNA environment.

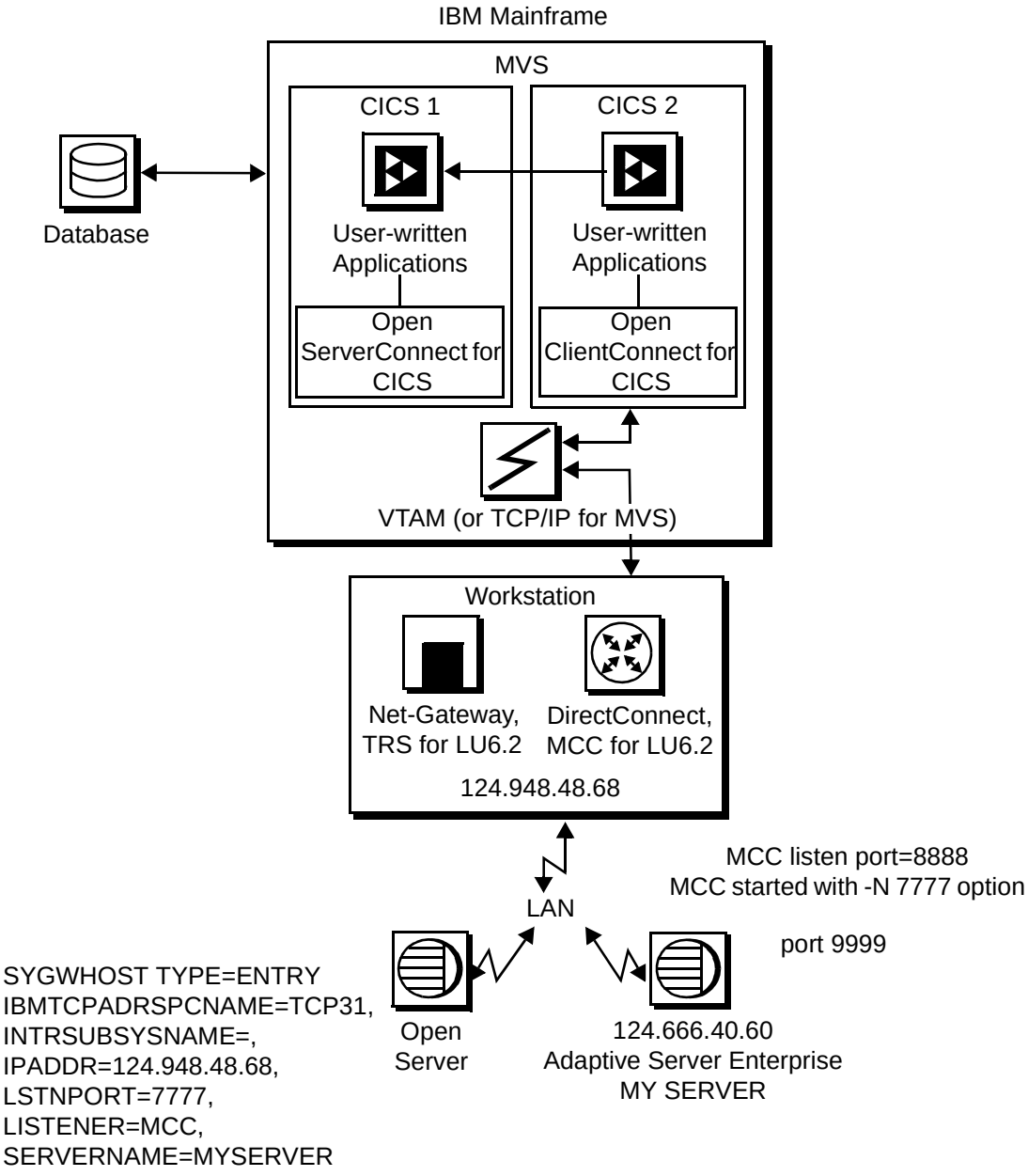
Figure 1-1: Open ClientConnect in a three-tier SNA environment



Three-tier TCP environment

Figure 1-2 shows Open ClientConnect communication in a three-tier (gateway-enabled) TCP environment.

Figure 1-2: Open ClientConnect in a three-tier TCP environment

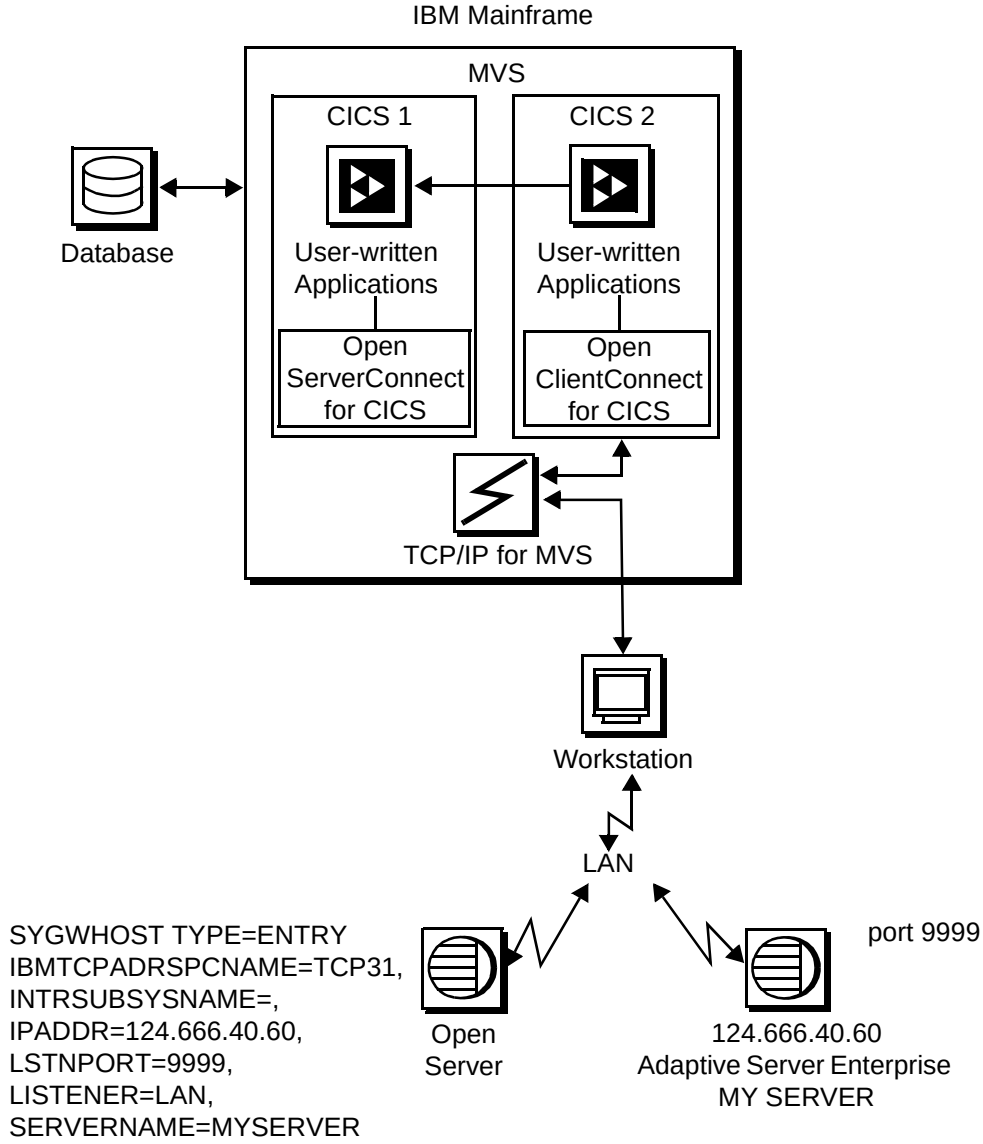


Two-tier (gateway-less) environment

Two-tier TCP environment

Figure 1-3 shows Open ClientConnect communication in a two-tier (gateway-less) TCP environment.

Figure 1-3: Open ClientConnect in a two-tier TCP environment



Communication flow

This section describes what happens at the mainframe, at the DirectConnect installation, and at the server in Open ClientConnect processing.

At the mainframe

An Open ClientConnect application calls a pre-written procedure, such as a stored procedure or an Open ServerConnect application. All calls from Open ClientConnect to remote nodes are processed using the LU 6.2 (three-tier only) or TCP/IP (two-tier only) communications protocol. For requests to an Open Server, the client can access any data available to the Open Server application. If the request is to Open ServerConnect, the client can access any data storage system accessible through CICS.

The called procedure or transaction executes and returns results to the calling Open ClientConnect application, which can use the results for local processing. If the client has permission, the client transaction can update data at remote sites by inserting, modifying, and deleting entries in database tables or other data storage systems. For more information on any of the following topics, refer to the Mainframe Connect Client Option *Installation and Administration Guide*.

isql utility

Open ClientConnect includes isql, a utility that allows users to send SQL language commands interactively. Users specify the server, whether or not to enable tracing, and type SQL commands in a 3270 panel.

Connection Router Table (SNA Only)

For SNA environments, Open ClientConnect includes a Connection Router Table that allows you to define servers and connections, and to set traffic priorities.

Server-Host Mapping Table (TCP/IP only)

For TCP/IP, Open ClientConnect includes a Server-Host Mapping Table that allows you to define servers for both three-tier and two-tier environments.

Side Information (SNA only)

Open ClientConnect for CICS uses the APPC Side Information File to define servers.

At the server

Typically, a server accepts requests from a client and returns results. In an Open ClientConnect environment, the server can be an Adaptive Server Enterprise, an Open Server™, or Open ServerConnect™ on the mainframe.

From the server standpoint, a request from an IBM host is no different than a request from a Sybase client. Open ClientConnect participates in ASCII-EBCDIC translations and datatype conversions.

Open ClientConnect security

Security for Open ClientConnect processing can be configured to require permission to:

- Log into the target server or desired CICS region.
- Use specific commands, stored procedures or transactions, and data objects at the target server.

For more security-related information regarding:

- Adaptive Server Enterprise, refer to the chapter called “Security Administration,” in the Adaptive Server Enterprise *System Administration Guide*.
- DirectConnect, refer to the Mainframe Connect DirectConnect for z/OS Option *User's Guide for Transaction Router Services*.
- Mainframes, refer to documentation provided with CICS and MVS, or the appropriate mainframe security system.

How to choose a network driver

Open ClientConnect supports concurrent use of multiple network drivers, providing additional flexibility and ease of installation for sites configured to run mixtures of SNA and TCP/IP.

Note Dynamic network driver support is a new feature in Open ClientConnect version 4.0.

The network drivers can be invoked from the same Open Client and Open Server common code base. The appropriate network driver is loaded dynamically at the time the program executes.

You must use the SYGWDRIV macro to define the network drivers to be used with Open ClientConnect and Open ServerConnect. For each operating environment (CICS and MVS), the default SYGWXCPH member provided on the tape contains the SYGWDRIV macro definitions for *all* the supported network drivers pertinent to the technology. The person installing Open ClientConnect should edit the appropriate SYCTCUST member to comment-out the drivers that your site does not intend to use.

This section provides an overview of network communication, lists and describes general criteria for choosing a driver, and explains how to choose between a CPI-C/LU 6.2 and a TCP/IP driver.

Overview of network communication definitions

You need to choose which dynamic network drivers to use at your site. Your choice depends on the protocols installed at your site and the types of processing you want to achieve.

Use this topic overview to understand issues involved in selecting your drivers:

- System Application Architecture (SAA)
- Common Programming Interface (CPI)
- APPC/MVS
- Systems Network Architecture (SNA)
- LU 6.2
- Advanced Program-to-Program Communications (APPC)
- Common threads between APPC MVS, CICS, and IMS TM

System Application Architecture (SAA)

SAA is an architecture composed of a set of selected software interfaces, conventions, and protocols designed to provide a framework for developing distributed applications. The key benefits of SAA are portability, consistency, and connectivity. The components of SAA are specifications for the key application interfaces points:

- Common user access
- Common communication support
- Common Programming Interface (CPI)

Common Programming Interface (CPI)

The SAA Common Programming Interface specifies the languages and services used to develop applications across SAA environments. The elements of the CPI specification are divided into two parts:

- Processing logic
  - High level language—COBOL, C, Fortran, RPG
  - Procedure language—REXX
  - Application generator—Cross Systems Product/Application Development (CSP/AD)
- Services
  - Communication interface or CPI-C—API for writing APPC applications.
  - Database interface—Structured Query Language (SQL)
  - Dialog interface—Interactive System Productivity Facility (ISPF)

APPC/MVS

APPC/MVS is an SNA application that extends APPC support to the MVS operating system. The primary role of APPC/MVS is to provide full LU 6.2 capability to MVS applications to allow communication with other applications in a distributed SNA network.

APPC/MVS provides programming support by providing an API based on the CPI-C interface. This interface is implemented in a lower-level API that is MVS-specific. The CPI-C calls all begin with CM; for example, CMALLC (Allocate). The MVS calls all begin with ATB; for example, Send_data (ATBSEND). The CPI-C calls are portable to non-MVS platforms while the ATB calls are not portable to non-MVS platforms.

Systems Network Architecture (SNA)

SNA is an IBM Network Architecture composed of a set of software interfaces, protocols, and operational sequences for transmitting information through and controlling the configuration and operation of networks.

LU 6.2

LU 6.2 refers to the SNA Logical Unit Type 6.2, which supports general communication between programs in a distributed environment. LU 6.2 is characterized by peer-to-peer communications support, comprehensive end-to-end error processing, optimized data transmission flow and a generic API.

The LU 6.2 system is layered functionally. It can be represented by a set of finite-state machines. Each of these machines has a finite number of states and a set of rules that govern the transition from one state to another. These finite state machines govern the behavior of LU 6.2 devices by guaranteeing that a given input always produces the same output.

Advanced Program-to-Program Communications (APPC)

APPC is peer level data communication support based on the SNA LU 6.2 protocols.

Common threads between APPC MVS, CICS, and IMS TM

All inbound transactions require a scheduler. Under MVS, the ASCH address space performs this function by scheduling inbound transactions in initiators under its control. The relationship between ASCH and its initiators is very similar to that of JES (Job Entry System), which schedules jobs in initiators under its control.

The Control region is the scheduler running under CICS. The Message Region running under the Control region corresponds to the initiators used by ASCH.

CICS differs from MVS and IMS TM because it does not schedule transactions in a separate address space. It schedules them as a task within its own address space.

Outbound transactions use a file called the Side Information File to map a name to an SNA logical unit. MVS and IMS TM both use this file.

General criteria for choosing a driver

The choice of a network driver depends on several factors:

- Network type—SNA or TCP/IP
- Network environment—two-tier (TCP/IP only) or three-tier (SNA only)
- Operating environment—CICS or MVS

This section explains why you might want to choose a particular driver in each environment.

Network type and environment

Because non-MVS platforms do not support LU 6.2 in their operating systems, if your network type is SNA, then you *must* use a three-tier network environment with a gateway. In a three-tier (gateway-enabled) environment, you must use either an LU 6.2 or CPI-C driver.

If your network type is TCP/IP, your network environment must be two-tier (gateway-less). The choice between IBM TCP/IP and Interlink depends on which product is installed in the network environment, although Open ClientConnect supports both concurrently.

Operating environment

This section explains the drivers used in CICS and MVS environments.

CICS environment

The following drivers are supported in the CICS environment:

- LU 6.2

- CPI-C
- IBM TCP/IP
- Interlink TCP/IP

The LU 6.2 driver only supports incoming transactions sent to the CICS message queue. It does not support Open Client outbound requests from the mainframe. With the LU 6.2 driver, CICS builds an entire result set in the message queue and sends that entire result set to the MSG.

The CPI-C driver supports both Open Client and Open Server requests. It does not use the CICS message queue to send or receive requests. Therefore, result sets sent by Open Server using the CPI-C driver can be interrupted. However, with the LU 6.2 driver, if a client does a CTRL-C to cancel the result set, the gateway must read the entire result set and throw it away.

MVS environment

The following drivers are supported in the MVS environment:

- CPI-C
- IBM TCP/IP
- Interlink TCP/IP

Choosing between a CPI-C/LU 6.2 driver and a TCP/IP driver

When choosing between a CPI-C/LU 6.2 driver and a TCP/IP driver, consider on the following factors:

- Network type - SNA or TCP/IP
- Reliability
- Performance
- Network operating environment - Two or three-tier

Network type

If your current network is SNA-only or TCP/IP-only, choose the driver that supports your network protocol.

Reliability

SNA networks have been running on IBM mainframes much longer than TCP/IP based-networks, and the SNA operational procedures are well established. However, if your LAN-side staff is not very familiar with SNA on a particular vendor platform, the SNA setup can be difficult.

TCP/IP is simpler to set up and maintain from the LAN, although it can be a challenge to get it running under MVS for the first time. In addition, TCP/IP on mainframes is a relatively new technology compared to SNA and as such, SNA is probably more robust and reliable.

Performance

TCP/IP performance appears to equal and in some cases exceed SNA-based performance. When going from the LAN to a mainframe, SNA requires a gateway, while TCP/IP does not.

Network operating environment

Small, two-tier (gateway-less) Client Server networks are easier to set up and maintain than three-tier (gateway-enabled) networks, because three-tier networks have a gateway between the mainframe and the LAN. However, three-tier networks scale better, as well as provide a single point of entry for security and tracing facilities that can be easily enabled.

Compatibility

For full functionality with the current version, use these mainframe access components listed in Table 1-1, as available at your site.

Table 1-1: Open ClientConnect version compatibility

Component	Version level
Open ServerConnect	3.1 or higher
Open ClientConnect	3.2 or higher
MainframeConnect for DB2 UDB	11.1 or higher
or OmniSQL Access Module for DB2	10.1 with latest Emergency Bug Fix (EBF) tapes
DirectConnect Transaction Router Service User's Guide	11.1
or Net–Gateway	3.0.1 for full functionality
Japanese Conversion Module	3.1

Open ClientConnect Client-Library functions

Open ClientConnect includes a programming interface of functions that are used in writing mainframe client applications. Some of these functions prepare and send requests to a server; others retrieve and process the results. Additional functions set application properties, handle error conditions, and provide a variety of information about an application's interaction with a server.

Open ClientConnect's Client-Library functions are similar in name and function to Open Client Client-Library's routines. However, not all Sybase Open Client routines are supported. For example, Open ClientConnect does not currently support cursors or compute rows.

Most mainframe Client-Library function names begin with "ct_"; for example, the function that exits from Client-Library is named ct_exit. This corresponds to the Client-Library/C routine ct_exit.

Open ClientConnect includes utility functions which allocate, drop, and assign properties to context handles, and one function used in datatype conversion. These functions begin with the prefix: "cs_" instead of "ct_." This naming convention derives from related Sybase products, but it is irrelevant for Open ClientConnect. Use cs_xxxxx functions just as if they were ct_xxxxx functions.

Structures, types, and values used by Client-Library functions are defined in header files.

Client-Library functions are described in detail in Chapter 2, "Functions."

Using Client-Library functions

An application programmer writes a client program, adding calls to Client-Library functions to set up control structures, connect to servers, send commands, process results, and clean up. A Client-Library program is compiled, linked, and run in the same way as any other C program under CICS or IMS TM.

Note An application program can act as both client and server. Such a program, called a *mixed-mode* program, contains both Client-Library calls to send requests and Gateway-Library calls to accept and process requests. For more information, as well as an example of a mixed-mode program, see the Mainframe Connect Server Option *Programmer's Reference for COBOL*.

Basic control structures

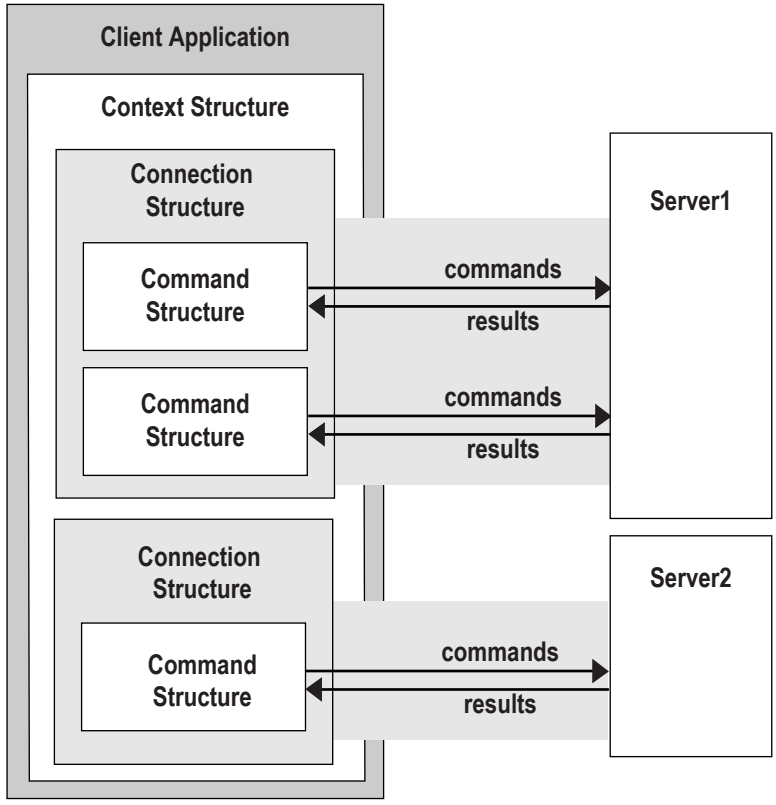
In order to send commands to a server, an application must allocate three types of structures:

- A context structure, which defines a particular application “context,” or operating environment
- A connection structure, which defines a particular client/server connection
- A command structure, which defines a “command space” in which commands are sent to a server

An application allocates these structures by calling the functions cs_ctx_alloc, ct_con_alloc, and ct_cmd_alloc.

The relationship between these control structures is illustrated in Figure 1-4 on page 17.

Figure 1-4: Client-Library's control structures



Through these structures, an application sets up its environment, connects to servers, sends commands, and processes results.

Note An Open ClientConnect application is restricted to one context per application. This differs from applications written in other versions of Client-Library, which support multiple context structures.

Steps in a simple program

A simple program involves the following steps:

1 Set up the programming environment:

cs_ctx_alloc — Allocate a context structure.
ct_init — Initialize the programming interface.

2 Establish a connection with a server or CICS or IMS region:

ct_con_alloc — Allocate a connection structure.
ct_con_props — Set or retrieve connection structure properties.
ct_connect — Connect to a server.

3 Send a command to the server or to a CICS or IMS region:
(For three-tier processing, send a command to Mainframe ClientConnect which forwards the request to the target server, and routes the results back to the client program).

ct_cmd_alloc — Allocate a command structure.
ct_command — Initiate a language request or remote procedure call.
ct_send — Send a request to the server.

4 Process the results of the command:

ct_results — Set up result data to be processed.
ct_res_info — Return result set information.
ct_bind — Bind a returned column or parameter to a program variable.
ct_fetch — Fetch result data.

5 Finish up:

ct_cmd_drop — Deallocate a command structure.
ct_close — Close a server connection.
ct_con_drop — Deallocate a connection structure.
ct_exit — Exit the programming interface.
cs_ctx_drop — Deallocate a context structure.

A simple language program

The following walk through demonstrates the basic framework of an Open ClientConnect application. The program follows the steps outlined in the previous section, sending a language request to an Adaptive Server and processing the results. In this case, the language command is a Transact-SQL select command.

Several working examples are provided on the product tape.

Note The *CTPUBLIC include* file is required in all source files that contain calls to Open ClientConnect.

Setting up the Client-Library programming environment

cs_ctx_alloc allocates a context structure. A context structure is used to store configuration parameters that describe a particular "context," or operating environment, for a set of connections.

Application properties that can be defined at the context level include the version of Client-Library being used, the login time-out value, and the maximum number of connections allowed within the context.

ct_init initializes your environment. It must be the first call in an application after cs_ctx_alloc.

Connecting to a server

ct_con_alloc allocates a connection structure. A connection structure contains information about a particular client/server connection.

ct_con_props sets and retrieves the property values of a connection. Connection properties include:

- User name and password, which are used in logging into a server
- Application name, which appears in Adaptive Server's sysprocesses table
- Packet size, which determines the size of network packets that an application sends and receives

- Dynamic network driver (LU6.2, IBM TCP/IP, Interlink TCP/IP, CPI-C), which defines the type of the network used between Open ClientConnect and the server

Open ClientConnect includes a Connection Router where servers and server connections are defined.

For a complete list of connection properties, see the section “Properties” on page 181 in Chapter 3, “Topics.”.

ct_connect opens a connection to a server, logging into the server with the connection information specified via ct_con_props.

Sending a command to the server

ct_cmd_alloc allocates a command structure. A command structure is used to send commands to a server and to process the results of those commands.

ct_command initiates the process of sending a command. In this example, it initiates a language command.

ct_send sends the command to the server.

Processing the results of the command

Almost all Client-Library programs process results by using a loop controlled by ct_results. Inside the loop, one of several actions takes place on the current type of result. Different types of results require different types of processing.

For row results, typically the number of columns in the result set is determined and then used to control a loop in which result items are bound to program variables. An application can call ct_res_info to get the number of result columns. After the result items are bound using ct_bind, the application calls ct_fetch to fetch data rows until end-of-data.

The results-processing model used in the example looks like this:

- Retrieve results: ct_results.
- Determine the type of results by examining the value in the result_type field.
- Process results.
  - If results are result rows: ct_res_info, ct_describe, ct_bind, ct_fetch.

- If results are return parameters: ct_param, ct_describe, ct_bind, ct_fetch.
- If results are status: ct_bind, ct_fetch.

ct_results sets up results for processing. The ct_results return parameter result_type indicates the type of result data that is available for processing.

Note that the example program calls ct_results in a loop that continues as long as ct_results returns CS_SUCCEED, indicating that result sets are available for processing. Although this type of program structure is not strictly necessary in the case of a simple language command, it is highly recommended. In more complex programs, it is not possible to predict the number and type of result sets than an application will receive in response to a command.

ct_bind binds a result item to a program variable. Binding creates an association between a result item and a program data space.

ct_fetch fetches result data. In the example, since binding has been specified and the count field in the DATAFMT structure for each column is set to 1, each ct_fetch call copies one row of data into program data space. As each row is fetched, the example program prints it.

After the ct_fetch loop terminates, the example program checks its final return code to find out whether it dropped out because of end-of-data, or because of failure.

Finishing up

Use the following functions to close a connection and deallocate the structures.

- ct_cmd_drop deallocates a command structure.
- ct_close closes a server connection.
- ct_con_drop deallocates a connection structure.
- ct_exit cleans up the remaining resources being used by command or connection handles.
- cs_ctx_drop deallocates a context structure.

CHAPTER 2

Functions

This chapter describes the functions that are included with your Open ClientConnect software. Table 2-1 provides a list of functions with a brief description of each one.

Table 2-1: List of functions

Function	*Description*
CTBBIND (see CTBBIND on page 25)	Binds a returned column or parameter to a program variable.
CTBCANCEL (see CTBCANCEL on page 35)	Cancels a request or the results of a request.
CTBCLOSE (see CTBCLOSE on page 37)	Closes a server connection.
CTBCMDALLOC (see CTBCMDALLOC on page 41)	Allocates a command handle.
CTBCMDDROP (see CTBCMDDROP on page 44)	Deallocates a command handle.
CTBCMDPROPS (see CTBCMDPROPS on page 47)	Sets, retrieves, or clears information about the current result set.
CTBCOMMAND (see CTBCOMMAND on page 51)	Initiates a language request or remote procedure call.
CTBCONALLOC (see CTBCONALLOC on page 55)	Allocates a connection handle.
CTBCONDROP (see CTBCONDROP on page 61)	Deallocates a connection handle.
CTBCONFIG (see CTBCONFIG on page 64)	Sets or retrieves context properties.
CTBCONNECT (see CTBCONNECT on page 67)	Connects to a server.
CTBCONPROPS (see CTBCONPROPS on page 70)	Sets or retrieves connection handle properties.
CTBDESCRIBE (see CTBDESCRIBE on page 78)	Returns a description of result data.
CTBDIAG (see CTBDIAG on page 86)	Manages in-line error handling.
CTBEXIT (see CTBEXIT on page 104)	Exits the programming interface.
CTBFETCH (see CTBFETCH on page 107)	Fetches result data.

Function	*Description*
CTBGETFORMAT (see CTBGETFORMAT on page 112)	Returns the server-defined format for a result column.
CTBINIT (see CTBINIT on page 117)	Initializes the programming interface.
CTBPARAM (see CTBPARAM on page 120)	Defines a command parameter.
CTBREMOTEPWD (see CTBREMOTEPWD on page 127)	Defines or clears passwords to be used for server-to-server connections.
CTBRESINFO (see CTBRESINFO on page 132)	Returns result set information.
CTBRESULTS (see CTBRESULTS on page 137)	Sets up result data to be processed.
CTBSEND (see CTBSEND on page 142)	Sends a request to the server.
CSBCONFIG (see CSBCONFIG on page 148)	Sets or retrieves global context properties.
CSBCONVERT (see CSBCONVERT on page 153)	Converts a data value from one datatype to another.
CSBCTXALLOC (see CSBCTXALLOC on page 160)	Allocates a context structure.
CSBCTXDROP (see CSBCTXDROP on page 162)	Deallocates a context structure.

CTBBIND

Description

Associates a returned column, parameter, or status with a program variable.

Syntax

```
COPY CTPUBLIC.
01 COMMAND       PIC S9(9) COMP SYNC.
01 RETCODE       PIC S9(9) COMP SYNC.
01 ITEM-NUM      PIC S9(9) COMP SYNC.
01 DATAFMT
   05 FMT-NAME          PIC X(132).
   05 FMT-NAMELEN       PIC S9(9) COMP SYNC.
   05 FMT-TYPE          PIC S9(9) COMP SYNC.
   05 FMT-FORMAT        PIC S9(9) COMP SYNC.
   05 FMT-MAXLEN        PIC S9(9) COMP SYNC.
   05 FMT-SCALE         PIC S9(9) COMP SYNC.
   05 FMT-PRECIS        PIC S9(9) COMP SYNC.
   05 FMT-STATUS        PIC S9(9) COMP SYNC.
   05 FMT-COUNT         PIC S9(9) COMP SYNC.
   05 FMT-UTYPE         PIC S9(9) COMP SYNC.
   05 FMT-LOCALE        PIC S9(9) COMP SYNC.
01 BUFFER
01 COPIED          PIC S9(9) COMP SYNC.
01 COPIED-NULL      PIC S9(9) COMP SYNC.
01 INDICATOR        PIC S9(4) COMP SYNC.
01 INDICATOR-NULL   PIC S9(9) COMP SYNC.

CALL 'CTBBIND' USING COMMAND RETCODE ITEM-NUM DATAFMT
BUFFER COPIED-NULL INDICATOR-NULL.
```

Parameters

COMMAND

(I) Handle for this connection. This is the handle defined in the CTBCMDALLOC call for this connection. The command handle corresponds to the *TDPROC* handle in the Open ServerConnect Gateway-Library.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under “Return value,” in this section.

ITEM-NUM

(I) Ordinal number of the result column, return parameter, or return status value that is to be bound.

When binding a result column:

- *ITEM-NUM* is the column number. For example, the first column in the select list of a SQL select statement is column number 1, the second is column number 2, and so forth.

When binding a return parameter:

- *ITEM-NUM* is the ordinal rank of the return parameter. The first parameter returned by a procedure or parameter is number 1. Adaptive Server stored procedure return parameters are returned in the order originally specified in the create procedure statement for the stored procedure. This is not necessarily the same order as specified in the RPC that invoked the stored procedure or transaction.

  In determining what number to assign to *ITEM-NUM*, do not count non-return parameters. For example, if the second parameter in a stored procedure is the only return parameter, its *ITEM-NUM* is 1.

When binding a stored procedure return status:

- *ITEM-NUM* must be 1. There is only one column and one row in a return status result set.

To clear all bindings:

- Assign *ITEM-NUM* a value of CS-UNUSED.

DATAFMT

(I) A structure that contains a description of the destination variable(s). This structure is also used by CTBDESCRIBE, CTBPARAM and CSBCONVERT and is explained in Chapter 3, “Topics”, under “DATAFMT structure” on page 170.

Table 2-2 on page 27 lists the fields in the *DATAFMT* structure, indicates whether they are used by CTBBIND, and contains general information about each field. CTBBIND ignores *DATAFMT* fields that it does not use.

Warning! You must initialize the entire *DATAFMT* structure to zeroes or low values. Failure to do so causes addressing exceptions.

Table 2-2: Fields in the DATAFMT structure for CTBBIND

Field	When used	Value represents
FMT-NAME	Not used (CS-FMT-UNUSED).	Not applicable.
FMT-NAMELEN	Not used (CS-FMT-UNUSED).	Not applicable.
FMT-TYPE	When binding all types of results.	The datatype of the destination variable (*BUFFER*). All datatypes listed under "Datatypes" on page 174 are valid. CTBBIND supports a wide range of datatype conversions, so FMT-TYPE can be different from the datatype returned by the server. For instance, by specifying a datatype of CS-FLOAT, you can bind a CS-MONEY or CS-MONEY4 value to a float-type program variable. The appropriate data conversion happens automatically. A return status always has a datatype of CS-INT.
FMT-FORMAT	When binding results to fixed-length character or binary destination variables. In all other cases, this field is unused (CS-FMT-UNUSED).	The destination format of character or binary data. For fixed-length character-type destinations only: CS-FMT-PADBLANK—pads to the full length of the variable with blanks. For fixed-length character or binary type destination variables: CS-FMT-PADNULL—pads to the full length of the variable with LOW-VALUES.
FMT-MAXLEN	When binding all types of results to non-fixed-length types. FMT-MAXLEN is ignored when binding to fixed-length datatypes.	The length of the destination variable, in bytes. If *BUFFER* has more than one element (that is, it is an array), FMT-MAXLEN is the length of one element. When binding to character or binary destinations, FMT-MAXLEN must describe the total length of the destination variable, including any space required for special terminating bytes, with this exception: when binding to a VARYCHAR-type destination such as DB2's VARCHAR, FMT-MAXLEN does not include the length of the "LL" length specification. To clear bind values, assign FMT-MAXLEN a value of 0. If the length specified in FMT-MAXLEN is too small to hold a result data item, then, at fetch time, CTBFETCH will discard the result item that is too large, fetch any remaining items in the row, and return CS-ROW-FAIL. If this occurs, the contents of *BUFFER* will be undefined. When binding Sybase-numerical/decimal to char, use CTDESCRIBE to determine precision. FMT-MAXLEN should be precision + 2 in this case. When binding to packed decimal CTBBIND calculates FMT-MAXLEN as (precision/2) + 1.

Field	When used	Value represents
FMT-SCALE	Only when converting column results or return parameters to or from an Open ServerConnect packed decimal (CS-PACKED370), Sybase-decimal, and Sybase-numeric datatypes.	The number of digits to the right of the decimal point. If the source value is the same datatype as the destination value, set FMT-SCALE to CS-SRC-VALUE to indicate that the destination variable should pick up the value for FMT-SCALE from the source data. FMT-SCALE must be less than or equal to FMT-PRECIS and cannot be greater than 31. If the actual scale is greater than the scale specified in FMT-SCALE but not greater than 31, CTBBIND truncates the results and issues a warning. If the actual scale is greater than 31, the CTBBIND call fails. When binding sybase-numeric/decimal to char or packed-decimal use CTDESCRIBE to determine precision and scale.
FMT-PRECIS	Only when converting column results or return parameters to an Open ServerConnect packed decimal (CS-PACKED370), Sybase-decimal, and Sybase-numeric datatypes.	The total number of decimal digits in the destination variable. This is the *n* in the *BUFFER* declaration: PIC S9(*n*)VG(*m*). If the source data is the same datatype as the destination variable, setting FMT-PRECIS to CS-SRC-VALUE instructs the destination variable to pick up its value for FMT-PRECIS from the source data. If the precision of the value fetched exceeds the precision of the destination variable, CTBFETCH returns a warning message. FMT-PRECIS must be greater than or equal to FMT-SCALE and cannot be less than 1 or greater than 31.
FMT-STATUS	Not used.	Not applicable.
FMT-COUNT	When binding all types of results. Only regular row result sets ever contain multiple rows. Other types of results (for example, return parameters, status) are treated like a single row of results.	The number of result rows to be copied to program variables per CTBFETCH call. If FMT-COUNT is larger than the number of available rows, only the available rows are copied. FMT-COUNT must have the same value for all columns in a result set according to the following: • If FMT-COUNT is 0 or 1, 1 row is fetched • If FMT-COUNT is greater than 1, it represents the number of rows that are fetched. In this case, *BUFFER* must be an array. **Note** Only regular row result sets can contain multiple rows. Other types of results (such as return parameters and status) are treated like a single row of results.
FMT-UTYPE	Not used.	Not applicable.
FMT-LOCALE	Not used.	Reserved for future use.

BUFFER

(I) Destination variable. A single field or an array of *n* elements where *n* is FMT-COUNT. Each array element is of size FMT-MAXLEN.

BUFFER is the program variable to which CTBBIND binds the server results. When the application calls CTBFETCH to fetch the result data, it is copied into this space.

The definition of the argument depends on the datatype of the destination variable. See Table 2-4 on page 33 for a list of possible values.

If you no longer want to store incoming data in this buffer, set FMT-MAXLEN to 0. This clears the binding.

COPIED

(O) Length of the incoming data. This can be a single field or, if *BUFFER* is an array, it can be an array of *n* elements where *n* is FMT-COUNT. At fetch time, CTBFETCH fills *COPIED* with the length(s) of the copied data.

COPIED-NULL

(I) *NULL* indicator for *COPIED*. This argument allows you to indicate that *COPIED* should be treated as null (LOW-VALUES). Assign this argument one of the following values:

Value	Meaning
CS-PARAM-NULL (-102)	*COPIED* is LOW-VALUES. If *COPIED* is an array, assigning CS-PARAM-NULL to this argument causes all elements of *COPIED* to be treated as LOW-VALUES.
CS-PARAM-NOTNULL (-103)	*COPIED* is not LOW-VALUES.

INDICATOR

-(O) From 1 to the value of FMT-COUNT integer variables. At fetch time, CTBFETCH uses each variable to indicate the following conditions about the fetched data:

Value	Integer value	Meaning
CS-NULLDATA	-1	There was no data to fetch. In this case, no data is copied to the destination variable.
CS-GOODDATA	0	The fetch was successful.
	n	The actual length of the server data, if the fetch resulted in truncation. *n* is an integer value.

If *BUFFER* is an array, *INDICATOR* will also be an array.

INDICATOR-NULL

(I) NULL indicator for *INDICATOR*. This argument allows you to treat *INDICATOR* as null (LOW-VALUES). Assign this argument one of the following values:

Value	Meaning
CS-PARAM-NULL (-102)	*INDICATOR* is LOW-VALUES. If *INDICATOR* is an array, assigning CS-PARAM-NULL to this argument causes all elements of *INDICATOR* to be treated as LOW-VALUES.
CS-PARAM-NOTNULL (-103)	INDICATOR is not LOW-VALUES.

Return value

CTBBIND returns one of the following values listed in Table 2-3.

Table 2-3: CTBBIND return values

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.
CS-FAIL (-2)	The routine failed.
TDS-CONNECTION-TERMINATED (-4997)	The connection is not active.
TDS-INVALID-DATAFMT-VALUE (-181)	DATAFMT field contains an illegal value.
TDS-INVALID-PARAMETER (-4)	A parameter was given an illegal value.
TDS-INVALID-VAR-ADDRESS (-175)	This value cannot be NULL.
TDS-NO-COMPUTES-ALLOWED (-60)	Compute results are not supported.
TDS-RESULTS-CANCELED (-49)	A cancel was sent to purge results.
TDS-SOS (-257)	Memory shortage. The operation failed.
TDS-WRONG-STATE (-6)	Program is in the wrong communication state to issue this call.

Examples

The following code fragment demonstrates the use of CTBBIND to set up column headings in result rows. It is taken from the sample program SYCTSAA5 in Appendix A, “Sample Language Requests.”

```
*=========================================================
*==                                                     ==
*== Subroutine to bind each data                        ==
*==                                                     ==
*=========================================================
     BIND-COLUMNS.

          CALL 'CTBDESCR' USING CSL-CMD-HANDLE,
                                CSL-RC,
                                I,
                                DATAFMT.
```

```
      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBDESCR failed'
                 DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.

**-------------------------------------------------------
**   We need TO bind the data TO program variables.
**   We don't care about the indicaTOr variable
**   so we'll pass NULL for that PARAMeter in OC-BIND().
**-------------------------------------------------------

******************
* ROWs per FETCH *
******************
       MOVE 1 TO DF-COUNT

       EVALUATE DF-DATATYPE

         WHEN CS-SMALLINT-TYPE

           CALL 'CTBBIND' USING CSL-CMD-HANDLE,
                                CSL-RC,
                                I,
                                DATAFMT,
                                DATA-SMALLINT,
                                CF-COL-LEN,
                                CS-PARAM-NOTNULL,
                                CF-COL-INDICATOR,
                                CS-PARAM-NULL

           IF CSL-RC NOT EQUAL CS-SUCCEED
             THEN
               MOVE SPACES TO MSGSTR
               STRING 'CTBBIND CS-SMALLINT-TYPE failed' DELIMITED
                                        BY SIZE INTO MSGSTR
               PERFORM PRINT-MSG
               PERFORM ALL-DONE
           END-IF

         WHEN CS-VARCHAR-TYPE

           MOVE LENGTH OF CF-COL-FIRSTNME-TXT TO DF-MAXLENGTH
```

```
                CALL 'CTBBIND' USING CSL-CMD-HANDLE,
                                     CSL-RC,
                                     I,
                                     DATAFMT,
                                     CF-COL-FIRSTNME,
                                     CF-COL-LEN,
                                     CS-PARAM-NOTNULL,
                                     CF-COL-INDICATOR,
                                     CS-PARAM-NULL

                IF CSL-RC NOT EQUAL CS-SUCCEED
                  THEN
                    MOVE SPACES TO MSGSTR
                    STRING 'CTBBIND CS-VARCHAR-TYPE failed' DELIMITED
                                                BY SIZE INTO MSGSTR
                    PERFORM PRINT-MSG
                    PERFORM ALL-DONE
                END-IF.

BIND-COLUMNS-EXIT.
    EXIT.
```

Usage

- CTBBIND associates (“binds”) a column, parameter, or status returned by a server to a program variable. Once a result is bound to a variable, any information returned in that column or parameter, or any status returned during a CTBFETCH call is copied to that variable.
- An application must call CTBBIND once for each result column or return parameter.
- CTBBIND can be used to bind a result column, a return parameter, or a stored procedure status value. When binding a result column, a single call to CTBBIND can bind multiple rows of the column. When binding a return status, you must bind a single variable of type integer.
- An application calls CTBBIND after CTBRESULTS and before CTBFETCH. CTBRESULTS tells the application whether there are any results to be bound and if so, what kind; CTBFETCH retrieves the results and copies them into the bound variable.
- CTBBIND binds only the current result type. CTBRESULTS indicates the current result type via its *RESULT-TYP* argument. For example, if CTBRESULTS returns CS-STATUS-RESULT, a return status is available for binding.

- An application can call CTBRESINFO to determine the number of items in the current result set, and can call CTBDESCRIBE to get a description of each item.
- An application can only bind a result item to a single program variable. If an application binds a result item to multiple variables, only the last binding takes effect.
- Binding for a particular type of result remains in effect until CTBRESULTS returns CS-CMD-DONE to indicate that the results of a logical command are processed completely.
- If you no longer want to store incoming data in the program variable, call CTBBIND with a zero-length *BUFFER* (for example, FMT-MAXLEN = 0).
- An application can rebind while actively fetching rows. That is, an application can call CTBBIND inside a CTBFETCH loop if it needs to change the binding of a result item (This action is not recommended).
- Table 2-4 lists the conversions performed by CTBBIND.

Table 2-4: Datatype conversions performed by CTBBIND

Source type	Result type
CS-VARCHAR	CS-CHAR
CS-CHAR	CS-VARCHAR
CS-MONEY	CS-CHAR
CS-MONEY	CS-VARCHAR
CS-FLT4	CS-FLT8
CS-MONEY	CS-FLT8
CS-PACKED370	CS-FLT8
CS-FLT8	CS-FLT4
CS-CHAR	CS-PACKED370
CS-VARCHAR	CS-PACKED370
CS-MONEY	CS-PACKED370
CS-FLT8	CS-PACKED370
CS-NUMERIC	CS-CHAR
CS-DECIMAL	CS-CHAR
CS-PACKED370	CS-DECIMAL
CS-NUMERIC	CS-PACKED370
CS-DECIMAL	CS-PACKED370
CS-DATETIME	CS-CHAR

Array binding

Array binding is the act of binding a result column to an array of program variables. At fetch time, multiple rows of the column are copied to the array of variables with a single CTBFETCH call. An application indicates array binding by assigning FMT-COUNT a value greater than 1

- Array binding is only practical for regular row results. Other types of results are considered to be the equivalent of a single row.
- When binding columns to arrays in a single command, all CTBBIND calls in the sequence of calls binding the columns must use the same value for FMT-COUNT. For example, when binding three columns to arrays, it is an error to assign FMT-COUNT a value of 5 in your first two CTBBIND calls and a value of 3 in the last.
- CTBBIND supports CS-NUMERIC and CS-DECIMAL datatypes.
- Use CTDESCRIBE before CTBBIND with decimal datatypes to get correct precision and scale.

See also

Related functions

- CTBDESCRIBE on page 78
- CTBFETCH on page 107
- CTBRESINFO on page 132
- CTBRESULTS on page 137

Related topics

- “Datatypes” on page 174

CTBCANCEL

Description

Cancels a request or the results of a request.

Syntax

```
01 CONNECTION PIC S9(9) COMP SYNC.
01 RETCODE PIC S9(9) COMP SYNC.
01 COMMAND PIC S9(9) COMP SYNC.
01 CANCELTYPE PIC S9(9) COMP SYNC.

CALL 'CTBCANCE' USING CONNECTION RETCODE
COMMAND CANCELTYPE.
```

Parameters

CONNECTION

(I) Handle for this connection. This connection handle must already be allocated with CTBCONALLOC. The connection handle corresponds to the *TDPROC* handle in the Open ServerConnect Gateway-Library.

Either *CONNECTION* or *COMMAND* must be null (LOW-VALUES). If *CONNECTION* is supplied and *COMMAND* is LOW-VALUES, the cancel operation applies to all commands pending for this connection.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under “Return values,” in this section.

COMMAND

(I) Handle for this client/server operation. This handle is defined in the associated CTBCMDALLOC call. The command handle also corresponds to the *TDPROC* handle in the Open ServerConnect Gateway-Library.

Either *CONNECTION* or *COMMAND* must be LOW-VALUES. If *COMMAND* is supplied and *CONNECTION* is LOW-VALUES, the cancel operation applies only to the command pending for this command structure.

CANCELTYPE

(I) Type of cancel requested. The following table lists the symbolic values that are legal for *CANCELTYPE*:

Value	Meaning
CS-CANCEL-ALL (6001) or CS-CANCEL-ATTN (6002)	CTBCANCEL sends an attention to the server, instructing it to cancel the current request, and immediately discards all results generated by the request.

Return value

CTBCANCEL returns one of the following values:

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.

Value	Meaning
CS-FAIL (-2)	The routine failed.
TDS-CONNECTION-TERMINATED (-4997)	The connection is not active.
TDS-INVALID-TDPROC (-18)	Specified command handle is invalid.
TDS-WRONG-STATE (-6)	Program is in the wrong communication state to issue this call.

Usage

- CTBCANCEL cancels the current result set.
- Canceling the current result set is equivalent to discarding the current set of results. Once results are discarded, they are no longer available to an application.
- In Open ClientConnect, CS-CANCEL-ALL and CS-CANCEL-ATTN function identically. Both immediately cancel the current request and discard all results generated by it.

Canceling a request

- To cancel the current request and all results generated by it, an application calls CTBCANCEL with *CANCELTYPE* as CS-CANCEL-ATTN or CS-CANCEL-ALL. These calls tell Client-Library to:
  - Discard all results already generated by the request.
  - Send an attention to the server instructing it to halt execution of the current request.

  For example, suppose the current request is a Transact-SQL language request that contains the queries:

```
select * from titles
select * from authors
```

- If an application cancels the language request after the first query executes but before the second query executes:
  - All remaining results from the first query are discarded.

- Execution of the second query is halted.

Note A call to CTBCANCEL with *CANCELTYPE* as CS-CANCEL-ALL or CS-CANCEL-ATTN must be immediately followed by a CTBRESULTS call.

- In Open Client Client-Library, canceling with *CANCELTYPE* as CS-CANCEL-ALL or CS-CANCEL-ATTN leaves the command structure in a "clean" state, available to be used for another operation.
- For both the CS-CANCEL-ATTN and CS-CANCEL-ALL types of cancels, if no request is in progress, CTBCANCEL returns CS-SUCCEED immediately.
- If a request initiates but has not been sent, a CS-CANCEL-ALL is rejected.

See also

Related functions

- CTBFETCH on page 107
- CTBRESULTS on page 137

CTBCLOSE

Description

Closes a server connection.

Syntax

```
COPY CTPUBLIC.

01 CONNECTION PIC S9(9) COMP SYNC.
01 RETCODE PIC S9(9) COMP SYNC.
01 OPTION PIC S9(9) COMP SYNC.

CALL 'CTBCLOSE' USING CONNECTION RETCODE OPTION.
```

Parameters

CONNECTION

(I) Handle for this connection. This connection handle must already be allocated with CTBCONALLOC. The connection handle corresponds to the *TDPROC* handle in the Open ServerConnect Gateway-Library.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under "Return value," in this section.

OPTION

(I) Option, if any, to use for the close. The following table lists the symbolic values that are legal for OPTION:

Value	Meaning
CS-UNUSED (-99999)	CTBCLOSE logs out and closes the connection. If the connection has results pending, CTBCLOSE returns CS-FAIL. This is the default behavior.
CS-FORCE-CLOSE (302)	CTBCLOSE closes the connection whether or not results are pending, and without notifying the server. This option is primarily for use when an application hangs waiting for a server response.
CS-KEEP-CON	This option is ignored. CICS treats it like CS-UNUSED.

Return value

CTBCLOSE returns one of the following values:

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.
CS-FAIL (-2)	The routine failed. The most common reason for a CTBCLOSE failure is pending results on the connection.
TDS-CONNECTION-TERMINATED (-4997)	The connection is not active.
TDS-COMMAND-ACTIVE (-7)	A command is in progress.
TDS-RESULTS-STILL-ACTIVE (-50)	Some results are still pending.

Examples

The following code fragment demonstrates the use of CTBCLOSE at the end of a program, after results processed. It is taken from the sample program SYCTSAA5 in Appendix A, “Sample Language Requests.”

```
*============================================================
*==                                                        ==
*== Subroutine to perform drop command handler, close     ==
*== server connection, and deallocate Connection Handler. ==
*==                                                        ==
*============================================================
    CLOSE-CONNECTION.
   ***************************
   * DROP THE COMMAND HANDLE *
   ***************************
        CALL 'CTBCMDDR' USING CSL-CMD-HANDLE
                              CSL-RC.
        IF CSL-RC = CS-FAIL
          THEN
            MOVE SPACES TO MSGSTR
```

```
            STRING 'CTBCMDDR failed' DELIMITED BY
                    SIZE INTO MSGSTR
            PERFORM PRINT-MSG
        END-IF.
*******************************
* CLOSE THE SERVER CONNECTION *
*******************************
        CALL 'CTBCLOSE' USING CSL-CON-HANDLE
                              CSL-RC
                              CS-UNUSED.
        IF CSL-RC = CS-FAIL
          THEN
            MOVE SPACES TO MSGSTR
            STRING 'CTBCLOSE failed' DELIMITED BY
                    SIZE INTO MSGSTR
            PERFORM PRINT-MSG
        END-IF.
***********************************
* DE-ALLOCATE THE CONNECTION HANDLE *
***********************************
        CALL 'CTBCONDR' USING CSL-CON-HANDLE
                              CSL-RC.
        IF CSL-RC = CS-FAIL
          THEN
            MOVE SPACES TO MSGSTR
            STRING 'CTBCONDR failed' DELIMITED BY
                    SIZE INTO MSGSTR
            PERFORM PRINT-MSG
        END-IF.
 CLOSE-CONNECTION-EXIT.
        EXIT.
```

Usage

- CTBCLOSE closes a server connection. All command handles associated with the connection are deallocated.
- To deallocate a connection handle, an application can call CTBCONDROP after the connection successfully closes.
- The behavior of CTBCLOSE depends on the value of *OPTION*, which determines the type of close. The following sections contain information on a type of close.

Default close behavior (OPTION is CS-UNUSED)

- If the connection has any pending results, CTBCLOSE returns CS-FAIL. To correct the failure, use CTBCLOSE with the CS-FORCE-CLOSE option or read in all of your results.

- Before terminating the connection with the server, CTBCLOSE sends a logout message to the server and reads the response to this message. The contents of this message do not affect the behavior of CTBCLOSE.

Forced close behavior (OPTION is CS-FORCE-CLOSE)

- The connection is closed whether or not it has pending results.
- Because this option sends no logout message to the server, the server cannot tell whether the close is intentional or whether it is the result of a lost connection or crashed client.

See also

Related functions

- CTBCONDROP on page 61
- CTBCONNECT on page 67
- CTBCONPROPS on page 70

CTBCMDALLOC

Description

Allocates a command handle.

Syntax

```
COPY CTPUBLIC.

01 CONNECTION   PIC S9(9) COMP SYNC.
01 RETCODE      PIC S9(9) COMP SYNC.
01 COMMAND      PIC S9(9) COMP SYNC.

CALL 'CTBCMDAL' USING CONNECTION RETCODE COMMAND.
```

Parameters

CONNECTION

(I) Handle for this connection. This connection handle must already be allocated with CTBCONALLOC. The connection handle corresponds to the *TDPROC* handle in the Open ServerConnect Gateway-Library.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under "Return value," in this section.

COMMAND

(O) Variable where this newly-allocated command handle is returned. All subsequent client requests using this connection must use this same name in the *COMMAND* argument. The command handle also corresponds to the *TDPROC* handle in the Open ServerConnect Gateway-Library.

In case of error, CTBCMDALLOC returns LOW-VALUES to this argument.

Return value

CTBCMDALLOC returns one of the following values listed in Table 2-5.

Table 2-5: CTBCMDALLOC return values

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.
CS-FAIL (-2)	The routine failed. The most common reason for a CTBCMDALLOC failure is a lack of adequate memory.
TDS-SOS (-257)	Memory shortage. The mainframe subsystem was unable to allocate enough memory for the control block that CTBCMDALLOC was trying to create. The operation failed.

Examples

The following code fragment demonstrates the use of CTBCMDALLOC. It is taken from the sample program SYCTSAA5 in Appendix A, "Sample Language Requests."

```
*============================================================
*==                                                        ==
*== Subroutine to allocate, send, and process commands ==
```

```
*==                                                          ==
*============================================================
      SEND-COMMAND.
  *---------------------------------------------------------------
  *  find out what the maximum number of connections is
  *---------------------------------------------------------------
       CALL 'CTBCONFI'  USING CSL-CTX-HANDLE,
                              CSL-RC,
                              CS-GET,
                              CS-MAX-CONNECT,
                              CF-MAXCONNECT,
                              CF-FOUR,
                              CS-FALSE,
                              CF-OUTLEN.
       IF CSL-RC NOT EQUAL CS-SUCCEED
         THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBCONFI CS-GET failed' DELIMITED BY SIZE
                                                  INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
       END-IF.
  *---------------------------------------------------------------
  *  allocate a command handle
  *---------------------------------------------------------------
       CALL 'CTBCMDAL' USING CSL-CON-HANDLE,
                             CSL-RC,
                             CSL-CMD-HANDLE.
       IF CSL-RC NOT EQUAL CS-SUCCEED
         THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBCMDAL failed' DELIMITED BY SIZE
                                                  INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
       END-IF.

  *---------------------------------------------------------------
  *    prepare the language request
  *---------------------------------------------------------------
       MOVE CF-LANG2-SIZE TO PF-STRLEN.
       CALL 'CTBCOMMA' USING CSL-CMD-HANDLE,
                             CSL-RC,
                             CS-LANG-CMD,
                             CF-LANG2,
                             PF-STRLEN,
```

```
                              CS-UNUSED.
           IF CSL-RC NOT EQUAL CS-SUCCEED
             THEN
               MOVE SPACES TO MSGSTR
               STRING 'CTBCOMMA CS-LANG-CMD failed' DELIMITED BY SIZE
                                                    INTO MSGSTR
               PERFORM PRINT-MSG
               PERFORM ALL-DONE
           END-IF.

      *-------------------------------------------------------------
      *    send the language request
      *-------------------------------------------------------------
           CALL 'CTBSEND' USING CSL-CMD-HANDLE,
                                CSL-RC.
           IF CSL-RC NOT EQUAL CS-SUCCEED
             THEN
               MOVE SPACES TO MSGSTR
               STRING 'CTBSEND failed' DELIMITED BY SIZE
                                                    INTO MSGSTR
               PERFORM PRINT-MSG
               PERFORM ALL-DONE
           END-IF.
       SEND-COMMAND-EXIT.
           EXIT.
```

Usage

- CTBCMDALLOC allocates a command handle on a specified connection. A command handle is a control structure that a Client-Library application uses to send requests to a server and process the results. Together, command and connection handles perform the functions of the Open ServerConnect *TDPROC* structure.
- Before calling CTBCMDALLOC, an application must allocate a connection structure via the Client-Library routine CTBCONALLOC.
- An application must call CTBCMDALLOC once for each logical command it issues. Each SQL statement is considered a separate logical command. For batched SQL, call CTBCMDALLOC once for each batch.

See also

Related functions

- CTBCMDDROP on page 44
- CTBCMDPROPS on page 47
- CTBCOMMAND on page 51
- CTBCONALLOC on page 55

Related documentation

- Mainframe Connect Client Option and Server Option *Messages and Codes*

CTBCMDDROP

Description

Deallocates a command handle.

Syntax

```
COPY CTPUBLIC.

01 COMMAND     PIC S9(9) COMP SYNC.
01 RETCODE     PIC S9(9) COMP SYNC.

CALL 'CTBCMDDR' USING COMMAND RETCODE.
```

Parameters

COMMAND

(I) Handle for this client/server operation. This handle is defined in the associated CTBCMDALLOC call. The connection handle corresponds to the *TDPROC* handle in the Open ServerConnect Gateway-Library.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under "Return value," in this section.

Return value

CTBCMDDROP returns one of the following values:

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.
CS-FAIL (-2)	The routine failed. *CTBCMDDROP* returns CS-FAIL if the command handle has any results pending.
TDS-COMMAND-ACTIVE (-7)	A command is in progress.
TDS-RESULTS-STILL-ACTIVE (-50)	Some results are still pending.

Examples

The following code fragment demonstrates the use of CTBCMDDROP at the end of a program, after results processed. It is taken from the sample program SYCTSAA5 in Appendix A, "Sample Language Requests."

```
*============================================================
*==                                                        ==
*== Subroutine to perform drop command handler, close     ==
*== server connection, and deallocate Connection Handler. ==
*==                                                        ==
*============================================================
     CLOSE-CONNECTION.
```

```
*****************************
* DROP THE COMMAND HANDLE *
*****************************
     CALL 'CTBCMDDR' USING CSL-CMD-HANDLE
                           CSL-RC.
     IF CSL-RC = CS-FAIL
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBCMDDR failed' DELIMITED BY
                 SIZE INTO MSGSTR
         PERFORM PRINT-MSG
     END-IF.
*******************************
* CLOSE THE SERVER CONNECTION *
*******************************
     CALL 'CTBCLOSE' USING CSL-CON-HANDLE
                           CSL-RC
                           CS-UNUSED.
      IF CSL-RC = CS-FAIL
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBCLOSE failed' DELIMITED BY
                 SIZE INTO MSGSTR
         PERFORM PRINT-MSG
     END-IF.
***********************************
* DE-ALLOCATE THE CONNECTION HANDLE *
***********************************
     CALL 'CTBCONDR' USING CSL-CON-HANDLE
                           CSL-RC.
     IF CSL-RC = CS-FAIL
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBCONDR failed' DELIMITED BY
                 SIZE INTO MSGSTR
         PERFORM PRINT-MSG
     END-IF.
 CLOSE-CONNECTION-EXIT.
     EXIT.
```

Usage

- CTBCMDDROP deallocates a command handle.
- If CTBCMDDROP is called while a command is pending (results have not all been returned), it fails. Before deallocating a command structure, an application should process or cancel any pending results.

- Once a command handle is deallocated, it cannot be reused. To allocate a new command handle, an application calls CTBCMDALLOC.

See also

Related functions

- CTBCMDALLOC on page 41
- CTBCOMMAND on page 51

CTBCMDPROPS

Description

Sets, retrieves, or clears information about the current result set.

Syntax

```
COPY CTPUBLIC.

01 COMMAND PIC S9(9) COMP SYNC.
01 RETCODE PIC S9(9) COMP SYNC.
01 ACTION  PIC S9(9) COMP SYNC.
01 PROPERTY PIC S9(9) COMP SYNC.
01 BUFFER type.
01 BUFFER-LEN  PIC S9(9) COMP SYNC.
01 BUFBLANKSTRIP PIC S9(9) COMP SYNC.
01 OUTLEN  PIC S9(9) COMP SYNC.

CALL 'CTBCMDPR' USING COMMAND RETCODE ACTION PROPERTY
BUFFER-LEN BUFBLANKSTRIP OUTLEN.
```

Parameters

COMMAND

(I) Handle for this client/server operation. This handle is defined in the associated CTBCMDALLOC call. The command handle corresponds to the *TDPROC* handle in the Open ServerConnect Gateway-Library.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under “Return value,” in this section.

ACTION

(I) Action to be taken by this call. *ACTION* is an integer variable that indicates the purpose of this call.

Assign *ACTION* one of the following symbolic values

Value	Meaning
CS-GET (33)	Retrieves the value of the property.
CS-SET (34)	Sets the value of the property.
CS-CLEAR (35)	Clears the value of the property by resetting the property to its Client-Library default value.

PROPERTY

(I) Symbolic name of the property for which the value is being set or retrieved. Client-Library properties are listed under “Properties” on page 181, with descriptions, possible values, and defaults.

BUFFER

(I/O) Variable (buffer) that contains the specified property value.

If *ACTION* is CS-SET, the buffer contains the value used by CTBCMDPROPS.

If *ACTION* is CS-GET, CTBCMDPROPS returns the requested information to this buffer.

If *ACTION* is CS-CLEAR, the buffer is reset to the default property value.

This argument is typically one of the following datatypes:

```
PIC S9(9) COMP SYNC.
PIC X(n).
```

BUFFER-LEN

(I) Length, in bytes, of the buffer.

If *ACTION* is CS-SET and the value in the buffer is a fixed-length or symbolic value, *BUFFER-LEN* should have a value of CS-UNUSED. To indicate that the terminating character is the last non-blank character, set *BUFBLANKSTRIP* to CS-TRUE.

If *ACTION* is CS-GET and *BUFFER* is too small to hold the requested information, CTBCMDPROPS sets *OUTLEN* to the length of the requested information and returns CS-FAIL. To retrieve all the requested information, change the value of *BUFFER-LEN* to the length returned in *OUTLEN* and rerun the application.

If *ACTION* is CS-CLEAR, set this value to CS-UNUSED.

BUFBLANKSTRIP

(I) Blank stripping indicator. Indicates whether or not trailing blanks are stripped.

Assign this argument one of the following symbolic values:

Value	Meaning
CS-TRUE (1)	Trailing blanks are stripped. The value in the buffer ends at the last non-blank character.
CS-FALSE (0)	Trailing blanks are not stripped. They are included in the value.

If a property value is being set and the terminating character is the last non-blank character, assign CS-TRUE to *BUFBLANKSTRIP*.

OUTLEN

(O) Length, in bytes, of the retrieved information. *OUTLEN* is an integer variable where CTBCMDPROPS returns the length of the property value being retrieved.

When the retrieved information is larger than *BUFFER-LEN* bytes, an application uses the value of *OUTLEN* to determine how many bytes are needed to hold the information.

OUTLEN is used only when *ACTION* is CS-GET. When *ACTION* is CS-CLEAR or CS-SET, this value is ignored.

Return value

CTBCMDPROPS returns one of the following values:

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.
CS-FAIL (-2)	The routine failed.
TDS-INVALID-PARAMETER (-4)	One or more arguments were given illegal values.
TDS-CANNOT-SET-VALUE (-43)	This property cannot be set by the application.

Examples

The following code fragment demonstrates the use of CTBCMDPROPS. This sample is not part of any sample program, so the Working Storage section is included.

```
  PROCEDURE DIVISION.
01 CTX             PIC S9(9) COMP SYNC VALUE +0.
01 CON             PIC S9(9) COMP SYNC VALUE +0.
01 CMD             PIC S9(9) COMP SYNC VALUE +0.
01 RET             PIC S9(9) COMP SYNC VALUE +0.
01 RETCODE         PIC S9(9) COMP SYNC VALUE +0.
01 CMDSTR          PIC X(200).
01 PARM1           PIC X(3).
01 STRLEN          PIC S9(9) COMP SYNC.
01 OUTLEN          PIC S9(9) COMP SYNC.
01 USER-DATA       PIC X(30).
01 USER-BUF        PIC X(8).
01 I               PIC S9(9) COMP SYNC.
01 DISP-MSG.
   05 TEST-CASEPIC X(10) VALUE IS 'RPC SAMPLE'.
   05 FILLERPIC X(4) VALUE IS SPACES.
   05 MSG.
      10 SAMP-LIT         PIC X(3).
      10 SAMP-RC          PIC -ZZZ9.
      10 FILLER           PIC X(3) VALUE IS SPACES.
      10 MSGSTR           PIC X(40) VALUE IS SPACES.
   PROCEDURE DIVISION.
P0.
```

```
 * NOW GET A COMMAND HANDLE.
      MOVE ZERO TO CMD.
      CALL 'CTBCMDAL' USING CON RETCODE CMD.
      IF RETCODE NOT EQUAL CS-SUCCEED
           MOVE SPACES TO MSGSTR
           STRING 'CTBCMDAL FAILED' DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALLDONE.
 * SET COMMAND PROPERTIES.
      STRING 'userdata' DELIMITED BY SIZE INTO CMDSTR.
      MOVE 8 TO STRLEN.
      CALL 'CTBCMDPR'   USING CMD RETCODE CS-SET CS-USERDATA
                        USER-DATA  STRLEN CS-FALSE OUTLEN.
         [check return code]

     CALL 'CTBCMDPR'   USING CMD RETCODE CS-GET CS-USERDATA
                       USER-BUF  STRLEN CS-FALSE OUTLEN.
         [check return code]

IF USER-DATA NOT EQUAL USER-BUF
      MOVE SPACES TO MSGSTR
      STRING 'CTBCMDPR RETURNED THE WRONG USER VALUE'
             DELIMITED BY SIZE INTO MSGSTR
      PERFORM PRINT-MSG.
```

Usage

- CTBCMDPROPS sets or retrieves the values of properties of command handle structures.
- Command handle properties affect the behavior of an application at the command structure level.
- Some command handle properties default to the value of the property in the parent context. To find out which ones, see “Properties” on page 181.

See also

Related functions

- CTBCMDALLOC on page 41
- CTBCONFIG on page 64
- CTBCONPROPS on page 70
- CTBRESINFO on page 132

Related topics

- “Buffers” on page 165
- “Properties” on page 181

CTBCOMMAND

Description

Initiates a language request or remote procedure call (RPC).

Syntax

```
COPY CTPUBLIC.

01 COMMAND PIC S9(9) COMP SYNC.
01 RETCODE PIC S9(9) COMP SYNC.
01 REQTYPE PIC S9(9) COMP SYNC.
01 BUFFER type.
01 BUFFER-LEN PIC S9(9) COMP SYNC.
01 OPTION PIC S9(9) COMP SYNC.

CALL 'CTBCOMMA' USING COMMAND RETCODE REQTYPE BUFFER
BUFFER-LEN OPTION.
```

Parameters

COMMAND

(I) Handle for this client/server operation. This handle is defined in the associated CTBCMDALLOC call. The command handle corresponds to the *TDPROC* handle in the Open ServerConnect Gateway-Library.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under “Return value,” in this section.

REQTYPE

(I) Type of request to initiate. The following symbolic values are legal for *REQTYPE*:

When REQTYPE is	CTBCOMMAND initiates	BUFFER contains
CS-LANG-CMD (148)	A language request.	The text of the language request.
CS-RPC-CMD (149)	A remote procedure call.	The name of the remote procedure.

BUFFER

(I) Variable (buffer) that contains the language request or RPC name.

This argument is typically one of the following datatypes:

```
01 BUFFER   PIC S9(9) COMP SYNC.
01 BUFFER   PIC X(n).
```

BUFFER-LEN

(I) Length, in bytes, of the buffer.

If the value in the buffer is a fixed-length or symbolic value, assign *BUFFER-LEN* a value of CS-UNUSED.

OPTION
Option associated with this request, if any.

Currently, only RPCs take options. For language requests, assign *OPTION* a value of CS-UNUSED.

The following symbolic values are legal for *OPTION* when *REQTYPE* is CS-RPC-CMD:

Value	Meaning
CS-RECOMPILE (188)	Recompile the stored procedure before executing it.
CS-NORECOMPILE (189)	Do not recompile the stored procedure before executing it.
CS-UNUSED (-99999)	No options are assigned.

Return value

CTBCOMMAND returns one of the following values listed in Table 2-6 on page 52.

Table 2-6: CTBCOMMAND return values

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.
CS-FAIL (-2)	The routine failed.
TDS-CONNECTION-TERMINATED (-4997)	The connection is not active.
TDS-INVALID-PARAMETER (-4)	A parameter contains an illegal value.
TDS-WRONG-STATE (-6)	Program is in the wrong communication state to issue this call.

Examples

The following code fragment demonstrates the use of CTBCOMMAND. It is taken from the sample program SYCTSAA5 in Appendix A, “Sample Language Requests.”

```
*-------------------------------------------------------------
*  allocate a command handle
*-------------------------------------------------------------
       CALL 'CTBCMDAL' USING CSL-CON-HANDLE,
                             CSL-RC,
                             CSL-CMD-HANDLE.
       IF CSL-RC NOT EQUAL CS-SUCCEED
         THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBCMDAL failed' DELIMITED BY SIZE
                                                  INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
       END-IF.
```

```
*-------------------------------------------------------------
*    prepare the language request
*-------------------------------------------------------------
     MOVE CF-LANG2-SIZE TO PF-STRLEN.
     CALL 'CTBCOMMA' USING CSL-CMD-HANDLE,
                           CSL-RC,
                           CS-LANG-CMD,
                           CF-LANG2,
                           PF-STRLEN,
                           CS-UNUSED.
     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBCOMMA CS-LANG-CMD failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.
*-------------------------------------------------------------
*    send the language request
*-------------------------------------------------------------
     CALL 'CTBSEND' USING CSL-CMD-HANDLE,
                          CSL-RC.
     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBSEND failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.
 SEND-COMMAND-EXIT.
     EXIT.
```

Usage

- CTBCOMMAND initiates a language request or RPC. Initiating a request is the first step in sending it to a server.
- Sending a request to a server is a three step process. To send a request to a server, an application must:
  a Call CTBCOMMAND to initiate the request. CTBCOMMAND sets up internal structures that are used in developing a request stream to send to the server.
  b Call CTBPARAM to pass parameters for the request. An application must call CTBPARAM once for each parameter in the request.

c Call CTBSEND to send the request to the server.

Language requests

Language requests contain character strings that represent requests in a server's own language. For example, language requests to Adaptive Server can include any legal Transact-SQL command.

- A language request can be in any language, as long as the server to which it is directed can understand it. For example, Adaptive Server understands Transact-SQL, but requests to DB2 must use the DB2 version of SQL.
- If the language request string contains variables, an application can pass values for these variables by calling CTBPARAM once for each variable that the language string contains. A language request can have up to 255 parameters.
- Transact-SQL request variables must begin with a colon (:).

Remote Procedure Calls (RPCs)

RPCs instruct a server to execute a stored procedure or transaction on either itself or a remote server.

- If an application uses an RPC to execute a stored procedure or transaction that requires parameters, the application calls CTBPARAM once for each parameter the stored procedure or transaction requires.
- After sending an RPC with CTBSEND, an application can process the stored procedure or transaction results with CTBRESULTS and CTBFETCH. The functions CTBRESULTS and CTBFETCH are used to process both the result rows generated by the stored procedure or transaction and the return parameters and status, if any.

See also

Related functions

- CTBCMDALLOC on page 41
- CTBPARAM on page 120
- CTBSEND on page 142

Related topics

- "Remote procedure calls (RPCs)" on page 190

CTBCONALLOC

Description

Allocates a connection handle.

Syntax

```
COPY CTPUBLIC.

01 CONTEXT PIC S9(9) COMP SYNC.
01 RETCODE PIC S9(9) COMP SYNC.
01 CONNECTION PIC S9(9) COMP SYNC.

CALL 'CTBCONAL' USING CONTEXT RETCODE CONNECTION.
```

Parameters

CONTEXT

(I) A context structure. The context structure is defined in the program call CSBCTXALLOC. The context structure corresponds to the *IHANDLE* in the Open ServerConnect Gateway-Library.

If this value is invalid or nonexistent, CTBCONALLOC fails.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under "Return value," in this section.

CONNECTION

(O) Handle for this connection. All subsequent Client-Library calls using this connection must use this same name in their *CONNECTION* argument. The connection handle corresponds to the *TDPROC* handle in the Open ServerConnect Gateway-Library.

This is the same value used to define the connection to the Open ClientConnect Connection Table.

In case of error, CTBCONALLOC returns LOW-VALUES to this argument.

Return value

CTBCONALLOC returns one of the following values listed in Table 2-7.

Table 2-7: CTBCONALLOC return values

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.
CS-FAIL (-2)	The routine failed. The most common reason for a CTBCONALLOC failure is a lack of adequate memory.
TDS-SOS (-257)	Memory shortage. The mainframe subsystem was unable to allocate enough memory for the control block that CTBCONALLOC was trying to create. The operation failed.
TDS-GWLIB-NO-STORAGE (-17)	Could not get DSA for Gateway-Library.

Examples

The following code fragment demonstrates the use of CTBCONALLOC. It is taken from the sample program SYCTSAA5 in Appendix A, “Sample Language Requests”.

```
 *=========================================================
*==                                                       ==
*== Subroutine to process input data                      ==
*==                                                       ==
*=========================================================
     PROCESS-INPUT.
    *********************************
    * ALLOCATE A CONNECTION HANDLE. *
    *********************************
         MOVE ZERO TO CSL-CON-HANDLE.
         CALL 'CTBCONAL' USING CSL-CTX-HANDLE
                               CSL-RC
                               CSL-CON-HANDLE.
         IF CSL-RC NOT EQUAL CS-SUCCEED
           THEN
              MOVE SPACES TO MSGSTR
              STRING 'CTBCONAL failed' DELIMITED BY SIZE INTO MSGSTR
              PERFORM PRINT-MSG
              PERFORM ALL-DONE
         END-IF.

    ******************
    * SET THE USER ID *
    ******************
         CALL 'CTBCONPR' USING CSL-CON-HANDLE
                               CSL-RC
                               CS-SET
                               CS-USERNAME
                               PF-USER
                               PF-USER-SIZE
                               CS-FALSE
                               OUTLEN.
         IF CSL-RC NOT EQUAL CS-SUCCEED
           THEN
              MOVE SPACES TO MSGSTR
              STRING 'CTBCONPR for user-id failed' DELIMITED BY SIZE
                                                   INTO MSGSTR
              PERFORM PRINT-MSG
              PERFORM ALL-DONE
         END-IF.
    ******************
    * SET THE PASSWORD *
    ******************
```

```
     CALL 'CTBCONPR' USING CSL-CON-HANDLE
                           CSL-RC
                           CS-SET
                           CS-PASSWORD
                           PF-PWD
                           PF-PWD-SIZE
                           CS-FALSE
                           OUTLEN.
     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBCONPR for password failed' DELIMITED BY SIZE
                                                INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
     END-IF.
*********************
* SET THE TRAN NAME *
*********************
     IF PF-TRAN-SIZE IS NOT EQUAL TO ZEROES THEN
         CALL 'CTBCONPR' USING CSL-CON-HANDLE
                               CSL-RC
                               CS-SET
                               CS-TRANSACTION-NAME
                               PF-TRAN
                               PF-TRAN-SIZE
                               CS-FALSE
                               OUTLEN
         IF CSL-RC NOT EQUAL CS-SUCCEED
           THEN
              MOVE SPACES TO MSGSTR
              STRING 'CTBCONPR for TRAN name failed'
                     DELIMITED BY SIZE INTO MSGSTR
              PERFORM PRINT-MSG
              PERFORM ALL-DONE
           END-IF
     END-IF.
*******************************
* SET THE NET DRIVER PROPERTY *
*******************************
     IF PF-NETDRV = SPACES OR PF-NETDRV = 'LU62'                X
                           OR PF-NETDRV = 'lu62'
         MOVE CS-LU62 TO NETDRIVER
     ELSE
       IF PF-NETDRV = 'IBMTCPIP' OR PF-NETDRV = 'ibmtcpip'
         MOVE CS-TCPIP TO NETDRIVER
```

```
      ELSE
        IF PF-NETDRV = 'INTERLIN' OR PF-NETDRV = 'interlin'
          MOVE CS-INTERLINK TO NETDRIVER
      ELSE
        IF PF-NETDRV = 'CPIC' OR PF-NETDRV = 'cpic'
          MOVE CS-NCPIC TO NETDRIVER
      END-IF.
      IF PF-DRV-SIZE IS NOT EQUAL TO ZEROES THEN
          CALL 'CTBCONPR' USING CSL-CON-HANDLE
                                CSL-RC
                                CS-SET
                                CS-NET-DRIVER
                                NETDRIVER
                                CS-UNUSED
                                CS-FALSE
                                OUTLEN
          IF CSL-RC NOT EQUAL CS-SUCCEED
            THEN
               MOVE SPACES TO MSGSTR
               STRING 'CTBCONPR for network driver failed'
                      DELIMITED BY SIZE INTO MSGSTR
               PERFORM PRINT-MSG
               PERFORM ALL-DONE
            END-IF
      END-IF.
*******************************
* SET FOR MAINFRAME EXTRA INFO *
*******************************
      CALL 'CTBCONPR' USING CSL-CON-HANDLE
                            CSL-RC
                            CS-SET
                            CS-EXTRA-INF
                            CS-TRUE
                            CS-UNUSED
                            CS-FALSE
                            CS-UNUSED.
      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBCONPR for extra info failed'
                              DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
      END-IF.
*********************************
* SETUP retrieval of All Messages *
```

```
**********************************
     CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                          CSL-RC,
                          CS-UNUSED,
                          CS-INIT,
                          CS-ALLMSG-TYPE,
                          CS-UNUSED,
                          CS-UNUSED.
     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBDIAG CS-INIT failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.
*****************************************
* set the upper limit of number of messages *
*****************************************
     MOVE 5 TO PF-MSGLIMIT.
     CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                          CSL-RC,
                          CS-UNUSED,
                          CS-MSGLIMIT,
                          CS-ALLMSG-TYPE,
                          CS-UNUSED,
                          PF-MSGLIMIT.
     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBDIAG CS-MSGLIMIT failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.
************************
* CONNECT TO THE SERVER *
************************
     CALL 'CTBCONNE' USING CSL-CON-HANDLE
                           CSL-RC
                           PF-SERVER
                           PF-SERVER-SIZE
                           CS-FALSE.
     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
          MOVE SPACES TO MSGSTR
```

```
     STRING 'CTBCONNE failed' DELIMITED BY SIZE INTO MSGSTR
     PERFORM PRINT-MSG
     PERFORM ALL-DONE
END-IF.
IF NO-ERRORS
  THEN
    PERFORM SEND-COMMAND
END-IF.
```

Usage

- CTBCONALLOC allocates a connection handle to a Mainframe ClientConnect or another processing region (three-tier processing), or a Adaptive Server if using two-tier (gateway-less) processing.
- Before calling CTBCONALLOC, an application must:
  - Call CSBCTXALLOC to allocate a context structure.
  - Call CTBINIT to initialize Client-Library.
- Connecting to a server is a three-step process. To connect to a server, an application:
  a Calls CTBCONALLOC to obtain a connection handle.
  b Calls CTBCONPROPS to set the values of connection-specific properties, if desired.
  c Calls CTBCONNECT to create the connection and log into the server.
- All connections created within a context pick up default property values from the parent context. An application can override these default values by calling CTBCONPROPS to set property values at the connection level.
- An application can have multiple connections to one or more servers at the same time.

  For example, an application can simultaneously have two connections to the server “mars,” one connection to the server “venus,” and one connection to a separate transaction processing region named CICX3. The context property CS-MAX-CONNECT, set by CTBCONFIG, determines the maximum number of connections allowed per context.

  Each server connection requires a separate connection handle.
- In order to send requests to a server, one or more command handles must be allocated for a connection. CTBCMDALLOC allocates a command handle.

See also

Related functions

- CSBCTXALLOC on page 160
- CTBCLOSE on page 37
- CTBCMDALLOC on page 41
- CTBCONNECT on page 67
- CTBCONPROPS on page 70

CTBCONDROP

Description — Deallocates a connection handle.

Syntax

```
COPY CTPUBLIC.

01 CONNECTION PIC S9(9) COMP SYNC.
01 RETCODE PIC S9(9) COMP SYNC.

CALL 'CTBCONDR' USING CONNECTION RETCODE.
```

Parameters

CONNECTION
(I) Handle for this connection. This must be the same value specified in the CTBCONALLOC call that initialized this connection.

RETCODE
(O) Variable where the result from an executed function returns. Its value is one of the codes listed under "Return value," in this section.

Return value — CTBCONDROP returns one of the following values:

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.
CS-FAIL (-2)	The routine failed. The most common reason for a CTBCONDROP failure is that the connection is still open.
TDS-CONNECTION-TERMINATED (-4997)	The connection is not active.

Examples — The following code fragment demonstrates the use of CTBCONDROP at the end of a program, after results processed. It is taken from the sample program SYCTSAA5 in Appendix A, "Sample Language Requests."

```
*============================================================
*==                                                        ==
*== Subroutine to perform drop command handler, close     ==
```

```
    *== server connection, and deallocate Connection Handler. ==
    *==                                                        ==
    *============================================================
CLOSE-CONNECTION.
       ****************************
       * DROP THE COMMAND HANDLE *
       ****************************
            CALL 'CTBCMDDR' USING CSL-CMD-HANDLE
                                  CSL-RC.
            IF CSL-RC = CS-FAIL
              THEN
                MOVE SPACES TO MSGSTR
                STRING 'CTBCMDDR failed' DELIMITED BY
                        SIZE INTO MSGSTR
                PERFORM PRINT-MSG
            END-IF.
       *******************************
       * CLOSE THE SERVER CONNECTION *
       *******************************
            CALL 'CTBCLOSE' USING CSL-CON-HANDLE
                                  CSL-RC
                                  CS-UNUSED.
            IF CSL-RC = CS-FAIL
              THEN
                MOVE SPACES TO MSGSTR
                STRING 'CTBCLOSE failed' DELIMITED BY
                        SIZE INTO MSGSTR
                PERFORM PRINT-MSG
            END-IF.
       ***********************************
       * DE-ALLOCATE THE CONNECTION HANDLE *
       ***********************************
            CALL 'CTBCONDR' USING CSL-CON-HANDLE
                                  CSL-RC.
            IF CSL-RC = CS-FAIL
              THEN
                MOVE SPACES TO MSGSTR
                STRING 'CTBCONDR failed' DELIMITED BY
                        SIZE INTO MSGSTR
                PERFORM PRINT-MSG
            END-IF.
        CLOSE-CONNECTION-EXIT.
            EXIT.
       *====================================================
       *==                                                ==
       *== Subroutine to perform exit client library and ==
```

```
*== deallocate context structure.                 ==
*==                                               ==
*=================================================
 QUIT-CLIENT-LIBRARY.
***************************
* EXIT THE CLIENT LIBRARY *
***************************
     CALL 'CTBEXIT' USING CSL-CTX-HANDLE
                          CSL-RC
                          CS-UNUSED.
     IF CSL-RC = CS-FAIL
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBEXIT failed' DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
     END-IF.
***********************************
* DE-ALLOCATE THE CONTEXT STRUCTURE *
***********************************
     CALL 'CSBCTXDR' USING CSL-CTX-HANDLE
                           CSL-RC.
     IF CSL-RC = CS-FAIL
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CSBCTXDR failed' DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
     END-IF.
     EXEC CICS RETURN END-EXEC.
 QUIT-CLIENT-LIBRARY-EXIT.
     EXIT.
```

Usage

- CTBCONDROP deallocates a connection handle and all command handles associated with that connection.
- Once a connection handle is deallocated, it cannot be reused. To allocate a new connection handle, an application calls CTBCONALLOC.
- An application cannot deallocate a connection handle until the connection it represents successfully closes. To close a connection, an application calls CTBCLOSE.

See also

Related functions

- CTBCLOSE on page 37
- CTBCONALLOC on page 55
- CTBCONNECT on page 67

- CTBCONPROPS on page 70

CTBCONFIG

Description

Sets or retrieves context properties.

Syntax

```
COPY CTPUBLIC.

01 CONTEXT PIC S9(9) COMP SYNC.
01 RETCODE PIC S9(9) COMP SYNC.
01 ACTION PIC S9(9) COMP SYNC.
01 PROPERTY PIC S9(9) COMP SYNC.
01 BUFFER type.
01 BUFFER-LEN PIC S9(9) COMP SYNC.
01 BUFBLANKSTRIP PIC S9(9) COMP SYNC.
01 OUTLEN  PIC S9(9) COMP SYNC.

CALL 'CTBCONFI' USING CONTEXT RETCODE ACTION PROPERTY
BUFFER  BUFFER-LEN BUFBLANKSTRIP OUTLEN.
```

Parameters

CONTEXT

(I) A context structure. The context structure is defined in the program call CSBCTXALLOC. It corresponds to the *IHANDLE* structure in the Open ServerConnect Gateway-Library.

If this value is invalid or nonexistent, CTBCONFIG fails.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under “Return value,” in this section.

ACTION

(I) Action to be taken by this call. *ACTION* is an integer variable that indicates the purpose of this call.

Assign *ACTION* one of the following symbolic values:

Value	Meaning
CS-GET (33)	Retrieves the value of the property.
CS-SET (34)	Sets the value of the property.
CS-CLEAR (35)	Clears the value of the property by resetting the property to its Client-Library default value.

PROPERTY

(I) Symbolic name of the property for which the value is being set or retrieved. Client-Library properties are listed under “Properties” on page 181, with description, possible values, and defaults.

BUFFER

(I/O) Variable (buffer) that contains the specified property value.

If *ACTION* is CS-SET, the buffer contains the value used by CTBCONFIG.

If *ACTION* is CS-GET, CTBCONFIG returns the requested information to this buffer.

If *ACTION* is CS-CLEAR, the buffer is reset to the default property value.

This argument is typically one of the following datatypes:

```
01 BUFFER  PIC S9(9) COMP SYNC.
01 BUFFER  PIC X(n).
```

BUFFER-LEN

(I) Length, in bytes, of the buffer.

If *ACTION* is CS-SET and the value in the buffer is a fixed-length or symbolic value, *BUFFER-LEN* should have a value of CS-UNUSED. To indicate that the terminating character is the last non-blank character, an application sets *BUFBLANKSTRIP* to CS-TRUE.

If *ACTION* is CS-GET and *BUFFER* is too small to hold the requested information, CTBCMDPROPS sets *OUTLEN* to the length of the requested information and returns CS-FAIL. To retrieve all the requested information, change the value of *BUFFER-LEN* to the length returned in *OUTLEN* and rerun the application.

If *ACTION* is CS-CLEAR, this value is ignored.

BUFBLANKSTRIP

(I) Blank stripping indicator. Indicates whether trailing blanks are stripped.

Assign this argument one of the following symbolic values:

Value	Meaning
CS-TRUE (1)	Trailing blanks are stripped. The value in the buffer ends at the last non-blank character.
CS-FALSE (0)	Trailing blanks are not stripped. They are included in the value.

OUTLEN

(O) Length, in bytes, of the retrieved information. *OUTLEN* is an integer variable where CTBCONFIG returns the length of the property value being retrieved.

When the retrieved information is larger than *BUFFER-LEN* bytes, an application uses the value of *OUTLEN* to determine how many bytes are needed to hold the information.

OUTLEN is used only when *ACTION* is CS-GET. If the *ACTION* is CS-SET or CS-CLEAR, this value is ignored.

Return value

CTBCONFIG returns one of the following values:

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.
CS-FAIL (-2)	The routine failed.

Examples

The following code fragment demonstrates how to use CTBCONFIG to determine the maximum number of connections. It is taken from the sample program SYCTSAA5 in Appendix A, "Sample Language Requests."

```
*-------------------------------------------------------------
*  find out what the maximum number of connections is
*-------------------------------------------------------------
       CALL 'CTBCONFI'  USING CSL-CTX-HANDLE,
                              CSL-RC,
                              CS-GET,
                              CS-MAX-CONNECT,
                              CF-MAXCONNECT,
                              CF-FOUR,
                              CS-FALSE,
                              CF-OUTLEN.
       IF CSL-RC NOT EQUAL CS-SUCCEED
         THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBCONFI CS-GET failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
       END-IF.
```

Usage

- CSBCONFIG sets or retrieves the values of CS-EXTRA-INF and CS-VERSION. All other context properties are set or reset by CTBCONFIG. Context properties define aspects of Client-Library behavior at the context level.

- All connections created within a context pick up default property values from the parent context. An application can override these default values by calling CTBCONPROPS to set property values at the connection level.
- If an application changes context property values after allocating connections for the context, the existing connections do not recognize the new property values. Only new connections allocated after the new context property values are set use the new values as defaults.

See also

Related functions

- CTBCMDPROPS on page 47
- CTBCONNECT on page 67
- CTBCONPROPS on page 70
- CTBINIT on page 117

Related topics

- "Buffers" on page 165
- "Properties" on page 181

CTBCONNECT

Description

Connects to a server.

Syntax

```
COPY CTPUBLIC.

01 CONNECTION          PIC S9(9) COMP SYNC.
01 RETCODE             PIC S9(9) COMP SYNC.
01 SERVERNAME          PIC X(30).
01 SERVERNAME-LEN      PIC S9(9) COMP SYNC.
01 BUFBLANKSTRIP       PIC S9(9) COMP SYNC.

CALL 'CTBCONNE' USING CONNECTION RETCODE SERVERNAME
SERVERNAME-LEN BUFBLANKSTRIP.
```

Parameters

CONNECTION

(I) Handle for this connection. This connection handle must already be allocated with CTBCONALLOC. The connection handle corresponds to the *TDPROC* handle in the Open ServerConnect Gateway-Library.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under "Return value," in this section.

SERVERNAME

(I) Name of the connected server. For clients running SNA, this is the name by which the server is known to the Open ClientConnect Server Path Definition Table. For clients running TCP/IP without a gateway, this is the actual name of the Adaptive Server in the LAN interfaces file.

You must assign a value to this argument. If a server name is not specified, CTBCONNECT fails.

SERVERNAME-LEN

(I) Length, in bytes, of *SERVERNAME*. If the server name ends at the last non-blank character, assign CS-TRUE to *BUFBLANKSTRIP*.

BUFBLANKSTRIP

(I) Blank stripping indicator. Indicates whether trailing blanks are stripped.

Assign this argument one of the following symbolic values:

Value	Meaning
CS-TRUE (1)	Trailing blanks are stripped. The value in the buffer ends at the last non-blank character.
CS-FALSE (0)	Trailing blanks are not stripped. They are included in the value.

Return value

CTBCONNECT returns one of the following values listed in Table 2-8.

Table 2-8: CTBCONNECT return values

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.
CS-FAIL (-2)	The routine failed.
TDS-CRTABLE-UNAVAILABLE (-31)	The Connection Router table cannot be loaded.
TDS-INVALID-NAMELENGTH	The length specified for the server name is invalid.
TDS-EXCEED-MAX-CONN (-39)	The maximum number of connections is already open.
TDS-ROUTE-NOT-AVAILABLE (-34)	All paths are in use; no more paths are available.
TDS-ROUTE-NOT-FOUND (-33)	No such route is defined to the Connection Router.
TDS-NOT-INITIALIZED (-32)	The Connection Router has not initialized.
TDS-SERVER-NOT-FOUND (-30)	The specified server name could not be found in the Name Server.
TDS-WRONG-STATE (-6)	Program is in the wrong communication state to issue this call.

Examples

The following code fragment demonstrates the use of CTBCONNECT. It is taken from the sample program SYCTSAA5 in Appendix A, “Sample Language Requests.”

```
**************************
```

```
       * CONNECT TO THE SERVER *
       ************************
      CALL 'CTBCONNE' USING CSL-CON-HANDLE
                                  CSL-RC
                                  PF-SERVER
                                  PF-SERVER-SIZE
                                  CS-FALSE.
              IF CSL-RC NOT EQUAL CS-SUCCEED
                THEN
                   MOVE SPACES TO MSGSTR
                   STRING 'CTBCONNE failed' DELIMITED BY SIZE INTO MSGSTR
                   PERFORM PRINT-MSG
                   PERFORM ALL-DONE
              END-IF.
              IF NO-ERRORS
                THEN
                  PERFORM SEND-COMMAND
              END-IF.
```

Usage

- CTBCONNECT establishes a connection between a mainframe transaction processing region and a remote server. Information about the connection is stored in a connection handle, which uniquely identifies the connection.
- The remote server can be another transaction processing region or server (Adaptive Server, Open Server, and so on). For clients running SNA, the name in the Server Path Definition Table is the name of the remote region or server. For clients running TCP/IP, SYGWHOST (server name and IP address) is used in conjunction with either the MVS-side information file for the specific drive, or the CICS partner table.
- When it establishes a connection, CTBCONNECT sets up communication with the server, forwards login information, and communicates any connection-specific property information to the server.
- Because creating a connection involves sending login information, an application must define login parameters (server user ID and password) before calling CTBCONNECT. An application calls CTBCONPROPS to define login parameters.
- The maximum number of open connections per context is determined by the CS-MAX-CONNECT property (set by CTBCONFIG). The default maximum is 25 connections.
- There are two ways that an attempt to establish a connection can fail (assuming that the system is correctly configured):

- If the specified server machine (the machine on which the server resides) is running correctly and the network is running correctly, but no server is listening on the specified port, the specified server machine signals the client, through a network error, that the connection cannot be formed. Regardless of the login time-out value, the connection fails.
- If the machine on which the server resides is down, the server does not respond. Because "no response" is not considered to be an error, the network does not signal the client that an error occurred. However, if a login time-out period is set, a time-out error occurs when the client fails to receive a response within the set period.

- To close a connection, an application calls CTBCLOSE.
- You can find more information about defining servers and connections in the Mainframe Connect Client Option *Installation and Administration Guide*.

See also

Related functions

- CTBCLOSE on page 37
- CTBCONALLOC on page 55
- CTBCONDROP on page 61
- CTBCONFIG on page 64
- CTBCONPROPS on page 70
- CTBREMOTEPWD on page 127

Related topics

- "Buffers" on page 165

CTBCONPROPS

Description

Sets or retrieves connection handle properties.

Syntax

```
COPY CTPUBLIC.

01 CONNECTION     PIC S9(9) COMP SYNC.
01 RETCODE        PIC S9(9) COMP SYNC.
01 ACTION         PIC S9(9) COMP SYNC.
01 PROPERTY       PIC S9(9) COMP SYNC.
01 BUFFER         type.
```

```
01 BUFFER-LEN      PIC S9(9) COMP SYNC.
01 BUFBLANKSTRIP   PIC S9(9) COMP SYNC.
01 OUTLEN          PIC S9(9) COMP SYNC.

CALL 'CTBCONPR' USING CONNECTION RETCODE ACTION PROPERTY
BUFFER BUFFER-LEN BUFBLANKSTRIP OUTLEN
```

Parameters

CONNECTION

(I) Handle for this connection. This connection handle must already be allocated with CTBCONALLOC. The connection handle corresponds to the *TDPROC* handle in the Open ServerConnect Gateway-Library.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under “Return value,” in this section.

ACTION

(I) Action to be taken by this call. *ACTION* is an integer variable that indicates the purpose of this call.

Assign *ACTION* one of the following symbolic values:

Value	Meaning
CS-GET (33)	Retrieves the value of the property.
CS-SET (34)	Sets the value of the property.
CS-CLEAR (35)	Clears the value of the property by resetting the property to its Client-Library default value.

PROPERTY

(I) Symbolic name of the property for which the value is being set or retrieved. Client-Library properties are listed under “Properties” on page 181, with description, possible values, and defaults.

BUFFER

(I/O) Variable (buffer) that contains the specified property value.

If *ACTION* is CS-SET, the buffer contains the value used by CTBCMDPROPS.

If *ACTION* is CS-GET, CTBCMDPROPS returns the requested information to this buffer.

If *ACTION* is CS-CLEAR, the buffer is reset to the default property value.

This argument is typically one of the following datatypes:

```
01 BUFFER   PIC S9(9) COMP SYNC.
01 BUFFER   PIC X(n).
```

BUFFER-LEN

(I/O) Length, in bytes, of the buffer.

If *ACTION* is CS-SET and the value in the buffer is a fixed-length or symbolic value, *BUFFER-LEN* should have a value of CS-UNUSED. To indicate that the terminating character is the last non-blank character, an application sets *BUFBLANKSTRIP* to CS-TRUE.

If *ACTION* is CS-GET and *BUFFER* is too small to hold the requested information, CTBCMDPROPS sets *OUTLEN* to the length of the requested information and returns CS-FAIL. To retrieve all the requested information, change the value of *BUFFER-LEN* to the length returned in *OUTLEN* and rerun the application.

If *ACTION* is CS-CLEAR, this value is ignored.

BUFBLANKSTRIP

(I) Blank stripping indicator. Indicates whether trailing blanks are stripped.

Assign this argument one of the following symbolic values:

Value	Meaning
CS-TRUE (1)	Trailing blanks are stripped. The value in the buffer ends at the last non-blank character.
CS-FALSE (0)	Trailing blanks are not stripped. They are included in the value.

If you are setting a property value and the terminating character is the last non-blank character, assign CS-TRUE to *BUFBLANKSTRIP*.

OUTLEN

(O) Length, in bytes, of the retrieved information. *OUTLEN* is an integer variable where CTBCONPROPS returns the length of the property value being retrieved.

If the retrieved information is larger than *BUFFER-LEN* in bytes, an application uses the value of *OUTLEN* to determine how many bytes are needed to hold the information.

OUTLEN is used only when *ACTION* is CS-GET. If *ACTION* is CS-CLEAR or CS-SET, this value is ignored.

Return value

CTBCONPROPS returns one of the following values:

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.
CS-FAIL (-2)	The routine failed.

Value	Meaning
TDS-CANNOT-SET-VALUE (-43)	This property cannot be set by the application.
TDS-INVALID-PARAMETER (-4)	One or more arguments contain illegal values.

Examples

The following code fragment demonstrates the use of CTBCONPROPS. It is taken from the sample program SYCTSAA5 in Appendix A, "Sample Language Requests."

```
*=========================================================
*==                                                     ==
*== Subroutine to process input data                    ==
*==                                                     ==
*=========================================================
        PROCESS-INPUT.

       *********************************
       * ALLOCATE A CONNECTION HANDLE. *
       *********************************
            MOVE ZERO TO CSL-CON-HANDLE.
            CALL 'CTBCONAL' USING CSL-CTX-HANDLE
                                  CSL-RC
                                  CSL-CON-HANDLE.
             IF CSL-RC NOT EQUAL CS-SUCCEED
              THEN
                 MOVE SPACES TO MSGSTR
                 STRING 'CTBCONAL failed' DELIMITED BY SIZE INTO MSGSTR
                 PERFORM PRINT-MSG
                 PERFORM ALL-DONE
            END-IF.

       ********************
       * SET THE USER ID *
       ********************
            CALL 'CTBCONPR' USING CSL-CON-HANDLE
                                  CSL-RC
                                  CS-SET
                                  CS-USERNAME
                                  PF-USER
                                  PF-USER-SIZE
                                  CS-FALSE
                                  OUTLEN.
            IF CSL-RC NOT EQUAL CS-SUCCEED
              THEN
                 MOVE SPACES TO MSGSTR
```

```
           STRING 'CTBCONPR for user-id failed' DELIMITED BY SIZE
                                                INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
      END-IF.
********************
* SET THE PASSWORD *
********************
       CALL 'CTBCONPR' USING CSL-CON-HANDLE
                             CSL-RC
                             CS-SET
                             CS-PASSWORD
                             PF-PWD
                             PF-PWD-SIZE
                             CS-FALSE
                             OUTLEN.
      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBCONPR for password failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
      END-IF.
********************
* SET THE TRAN NAME *
********************
      IF PF-TRAN-SIZE IS NOT EQUAL TO ZEROES THEN
          CALL 'CTBCONPR' USING CSL-CON-HANDLE
                                CSL-RC
                                CS-SET
                                CS-TRANSACTION-NAME
                                PF-TRAN
                                PF-TRAN-SIZE
                                CS-FALSE
                                OUTLEN
          IF CSL-RC NOT EQUAL CS-SUCCEED
            THEN
               MOVE SPACES TO MSGSTR
               STRING 'CTBCONPR for TRAN name failed'
                      DELIMITED BY SIZE INTO MSGSTR
               PERFORM PRINT-MSG
               PERFORM ALL-DONE
            END-IF
      END-IF.
******************************
```

```
* SET THE NET DRIVER PROPERTY *
*******************************
     IF PF-NETDRV = SPACES OR PF-NETDRV = 'LU62'                X
                          OR PF-NETDRV = 'lu62'
         MOVE CS-LU62 TO NETDRIVER
     ELSE
       IF PF-NETDRV = 'IBMTCPIP' OR PF-NETDRV = 'ibmtcpip'
         MOVE CS-TCPIP TO NETDRIVER
     ELSE
       IF PF-NETDRV = 'INTERLIN' OR PF-NETDRV = 'interlin'
         MOVE CS-INTERLINK TO NETDRIVER
     ELSE
       IF PF-NETDRV = 'CPIC' OR PF-NETDRV = 'cpic'
         MOVE CS-NCPIC TO NETDRIVER
     END-IF.
     IF PF-DRV-SIZE IS NOT EQUAL TO ZEROES THEN
         CALL 'CTBCONPR' USING CSL-CON-HANDLE
                               CSL-RC
                               CS-SET
                               CS-NET-DRIVER
                               NETDRIVER
                               CS-UNUSED
                               CS-FALSE
                               OUTLEN
         IF CSL-RC NOT EQUAL CS-SUCCEED
           THEN
              MOVE SPACES TO MSGSTR
              STRING 'CTBCONPR for network driver failed'
                     DELIMITED BY SIZE INTO MSGSTR
              PERFORM PRINT-MSG
              PERFORM ALL-DONE
           END-IF
     END-IF.
*******************************
* SET FOR MAINFRAME EXTRA INFO *
*******************************
     CALL 'CTBCONPR' USING CSL-CON-HANDLE
                           CSL-RC
                           CS-SET
                           CS-EXTRA-INF
                           CS-TRUE
                           CS-UNUSED
                           CS-FALSE
                           CS-UNUSED.
     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
```

```
            MOVE SPACES TO MSGSTR
            STRING 'CTBCONPR for extra info failed'
                                DELIMITED BY SIZE INTO MSGSTR
            PERFORM PRINT-MSG
            PERFORM ALL-DONE
        END-IF.
```

Usage

- CTBCONPROPS sets or retrieves the values of properties for a connection handle. Connection properties define aspects of Client-Library behavior at the connection level.
- All command structures allocated for a connection pick up default property values from the parent connection. An application can override these default values by calling CTBCMDPROPS at the command structure level.
- If an application changes connection property values after allocating command structures for the connection, the existing command structures do not recognize the new property values. New command structures allocated for the connection use the new property values as defaults.
- Some connection properties only take effect if they are set before an application calls CTBCONNECT to establish the connection.
- An application can use CTBCONPROPS to set or retrieve the following properties:
  - CS-APPNAME
  - CS-CHARSETCNV
  - CS-COMMBLOCK
  - CS-EXTRA-INF
  - CS-HOSTNAME
  - CS-LOGIN-STATUS
  - CS-NET-DRIVER
  - CS-NETIO
  - CS-NOINTERRUPT
  - CS-PACKETSIZE
  - CS-PASSWORD
  - CS-TDS-VERSION

- CS-TRANSACTION-NAME
- CS-USERDATA

See also

Related functions

- CTBCMDPROPS on page 47
- CTBCONFIG on page 64
- CTBCONNECT on page 67
- CTBINIT on page 117

Related topics

- “Buffers” on page 165
- “Properties” on page 181

CTBDESCRIBE

Description

Returns a description of result data.

Syntax

```
COPY CTPUBLIC.

01 COMMAND       PIC S9(9) COMP SYNC.
01 RETCODE       PIC S9(9) COMP SYNC.
01 ITEM-NUM      PIC S9(9) COMP SYNC.
01 DATAFMT
05 FMT-NAME      PIC X(132).
05 FMT-NAMELEN   PIC S9(9) COMP SYNC.
05 FMT-TYPE      PIC S9(9) COMP SYNC.
05 FMT-FORMAT    PIC S9(9) COMP SYNC.
05 FMT-MAXLEN    PIC S9(9) COMP SYNC.
05 FMT-SCALE     PIC S9(9) COMP SYNC.
05 FMT-PRECIS    PIC S9(9) COMP SYNC.
05 FMT-STATUS    PIC S9(9) COMP SYNC.
05 FMT-COUNT     PIC S9(9) COMP SYNC.
05 FMT-UTYPE     PIC S9(9) COMP SYNC.
05 FMT-LOCALE    PIC S9(9) COMP SYNC.

CALL 'CTBDESCR' USING COMMAND RETCODE ITEM-NUM DATAFMT.
```

Parameters

COMMAND

(I) Handle for this client/server operation. This handle is defined in the associated CTBCMDALLOC call.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under “Return value,” in this section.

ITEM-NUM

(I) Ordinal number of the column, parameter, or status being returned. This value is an integer.

When describing a column, *ITEM-NUM* is the column number. For example, the first column in the select list of a SQL select statement is column number 1, the second is column number 2, and so forth.

When describing a return parameter, *ITEM-NUM* is the ordinal rank of the parameter. The first parameter returned by a procedure or transaction is number 1. Adaptive Server stored procedure return parameters are returned in the order originally specified in the stored procedure's create procedure statement. This is not necessarily the same order as specified in the RPC that invoked the stored procedure or transaction.

In determining what number to assign to *ITEM-NUM*, do not count non-return parameters. For example, if the second parameter in a stored procedure or transaction is the only return parameter, its *ITEM-NUM* is 1.

When describing a stored procedure return status, *ITEM-NUM* must be 1, because there can be only a single status in a return status result set.

To clear all bindings, assign *ITEM-NUM* a value of CS-UNUSED.

DATAFMT

(O) A structure that contains a description of the result data item referenced by *ITEM-NUM*. This structure is also used by CTBBIND, CTBPARAM and CSBCONVERT and is explained in the Topics chapter, under "DATAFMT structure" on page 170.

Warning! You must initialize DATAFMT to zeroes. Failure to do so causes addressing exceptions.

The DATAFMT structure contains the following fields listed in Table 2-9 on page 79.

Table 2-9: Fields in the DATAFMT structure for CTBDESCRIBE

When this field	Is used with these result items	CTBDESCRIBE sets the field to
FMT-NAME	Regular columns, return parameters	The null-terminated name of the data item, if any. To indicate that there is no name, set FMT-NAMELEN to 0.
FMT-NAMELEN	Regular columns, return parameters	The actual length, in bytes, of FMT-NAME, not including the null terminator. A zero value here indicates no name.

When this field	Is used with these result items	CTBDESCRIBE sets the field to
FMT-TYPE	Regular columns, return parameters, return status	The datatype of the data item. All “Datatypes” on page 174 are valid. A return status always has a datatype of CS-INT.
FMT-FORMAT	Not used (CS-FMT-UNUSED)	Not applicable.
FMT-MAXLEN	Regular columns, return parameters	The maximum possible length, in bytes, of the data for the column or parameter being described.
FMT-SCALE	Regular columns and return parameters for which the datatype is packed decimal (CS-PACKED370), or Sybase-decimal/numeric	The number of digits to the right of the decimal point.
FMT-PRECIS	Regular columns and return parameters for which the datatype is packed decimal (CS-PACKED370), or Sybase-decimal/numeric	The total number of decimal digits in the result data item.
FMT-STATUS	Regular columns only	One or more of the following symbolic values, added together: • CS-CANBENULL to indicate a column that was tagged “nullable” by the server. • CS-NODATA to indicate that no data is associated with the column.
FMT-COUNT	Regular columns, return parameters, return status	The number of rows copied to destination variables per CTBFETCH call. CTBDESCRIBE initializes FMT-COUNT as 1 to provide a default value in case an application uses the CTBDESCRIBE return DATAFMT structure as the CTBBIND input DATAFMT structure. This value is always 1 for return parameters and status results.
FMT-UTYPE	Regular columns, return parameters	The user-defined datatype of the column or parameter, if any. FMT-UTYPE is set in addition to (not instead of) DATATYPE. **Note** This field is used for datatypes defined at the server, not for Open Client user-defined datatypes.
FMT-LOCALE	Reserved for future use	Not applicable.

Return value

CTBDESCRIBE returns one of the following values listed in Table 2-10.

Table 2-10: CTBDESCRIBE return values

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.
CS-FAIL (-2)	The routine failed. CTBDESCRIBE returns CS-FAIL if *ITEM-NUM* does not represent a valid result data item.
TDS-CANCEL-RECEIVED (-12)	Operation canceled. The remote partner issued a cancel. The current operation failed.
TDS-CONNECTION-TERMINATED (-4997)	The connection is not active.
TDS-NO-COMPUTES-ALLOWED (-60)	Compute results are not supported.
TDS-RESULTS-CANCELED (-49)	A cancel was sent to purge results.
TDS-WRONG-STATE (-6)	Program is in the wrong communication state to issue this call.

Examples

The following code fragment demonstrates the use of CTBDESCRIBE. It is taken from the sample program SYCTSAA5 in Appendix A, "Sample Language Requests."

```
*=========================================================
*==                                                     ==
*== Subroutine to process result rows                   ==
*==                                                     ==
*=========================================================
 RESULT-ROW-PROCESSING.

     CALL 'CTBRESIN' USING CSL-CMD-HANDLE,
                           CSL-RC,
                           CS-NUMDATA,
                           RF-NUMDATA,
                           RF-NUMDATA-SIZE,
                           CF-COL-LEN.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBRESINFO failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

     COMPUTE FF-ROW-NUM = FF-ROW-NUM + 1.

*********************************
* display number of connections *
```

```
*********************************

     MOVE CF-MAXCONNECT   TO OR2-MAXCONNECT.
     MOVE OUTPUT-ROW-STR2 TO RSLTNO(FF-ROW-NUM).
     COMPUTE FF-ROW-NUM = FF-ROW-NUM + 2.

*********************************
* display the number of columns *
*********************************

     MOVE RF-NUMDATA      TO OR4-NUMDATA.
     MOVE OUTPUT-ROW-STR4 TO RSLTNO(FF-ROW-NUM).

     IF RF-NUMDATA NOT EQUAL 2
       THEN
         STRING 'CTBRESINFO returned wrong # of parms' DELIMITED
                                          BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

     COMPUTE FF-ROW-NUM = FF-ROW-NUM + 2.

**-------------------------------------------------------------
**   Setup column headings
**-------------------------------------------------------------

     MOVE 'FirstName    EducLvl' TO RSLTNO(FF-ROW-NUM).
     COMPUTE FF-ROW-NUM = FF-ROW-NUM + 1.
     MOVE '===========  =======' TO RSLTNO(FF-ROW-NUM).

     PERFORM BIND-COLUMNS
        VARYING I FROM 1 BY 1
           UNTIL I IS GREATER THAN RF-NUMDATA.

 RESULT-ROW-PROCESSING-EXIT.
     EXIT.

*=========================================================
*==                                                     ==
*== Subroutine to bind each data                        ==
*==                                                     ==
*=========================================================
 BIND-COLUMNS.

      CALL 'CTBDESCR' USING CSL-CMD-HANDLE,
```

```
                              CSL-RC,
                              I,
                              DATAFMT.

       IF CSL-RC NOT EQUAL CS-SUCCEED
         THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBDESCR failed'
                  DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
       END-IF.

**------------------------------------------------------
**   We need TO bind the data TO program variables.
**   We don't care about the indicaTOr variable
**   so we'll pass NULL for that PARAMeter in OC-BIND().
**------------------------------------------------------

******************
* ROWs per FETCH *
******************
       MOVE 1 TO DF-COUNT

       EVALUATE DF-DATATYPE

         WHEN CS-SMALLINT-TYPE

           CALL 'CTBBIND' USING CSL-CMD-HANDLE,
                                CSL-RC,
                                I,
                                DATAFMT,
                                DATA-SMALLINT,
                                CF-COL-LEN,
                                CS-PARAM-NOTNULL,
                                CF-COL-INDICATOR,
                                CS-PARAM-NULL

           IF CSL-RC NOT EQUAL CS-SUCCEED
             THEN
               MOVE SPACES TO MSGSTR
               STRING 'CTBBIND CS-SMALLINT-TYPE failed' DELIMITED
                                          BY SIZE INTO MSGSTR
               PERFORM PRINT-MSG
               PERFORM ALL-DONE
           END-IF
```

```
                WHEN CS-VARCHAR-TYPE

                  MOVE LENGTH OF CF-COL-FIRSTNME-TXT TO DF-MAXLENGTH

                  CALL 'CTBBIND' USING CSL-CMD-HANDLE,
                                       CSL-RC,
                                       I,
                                       DATAFMT,
                                       CF-COL-FIRSTNME,
                                       CF-COL-LEN,
                                       CS-PARAM-NOTNULL,
                                       CF-COL-INDICATOR,
                                       CS-PARAM-NULL

                  IF CSL-RC NOT EQUAL CS-SUCCEED
                    THEN
                      MOVE SPACES TO MSGSTR
                      STRING 'CTBBIND CS-VARCHAR-TYPE failed' DELIMITED
                                                 BY SIZE INTO MSGSTR
                      PERFORM PRINT-MSG
                      PERFORM ALL-DONE
                  END-IF.

            BIND-COLUMNS-EXIT.
                EXIT.
```

Usage

- CTBDESCRIBE returns a complete description of a result data item in the current result set. Result data items include regular result columns, return parameters, and stored procedure return status values.
- An application can call CTBRESINFO to find out how many result items are present in the current result set.
- An application generally needs to call CTBDESCRIBE to describe a result data item after it establishes a connection and sends a request, and before it binds the result item to a program variable using CTBBIND.
- CTBDESCRIBE also indicates when the client issues a cancel.

See also

Related functions

- CTBBIND on page 25
- CTBFETCH on page 107
- CTBRESINFO on page 132

- CTBRESULTS on page 137

Related topics

- “DATAFMT structure” on page 170
- “Results” on page 193

CTBDIAG

Description

Manages in-line error handling.

Syntax

```
COPY CTPUBLIC.

01 CONNECTION      PIC S9(9) COMP SYNC.
01 RETCODE         PIC S9(9) COMP SYNC.
01 COMPILER        PIC S9(9) COMP SYNC.
01 OPERATION       PIC S9(9) COMP SYNC.
01 MSGTYPE         PIC S9(9) COMP SYNC.
01 INDEX           PIC S9(9) COMP SYNC.
01 BUFFER          type.

CALL 'CTBDIAG' USING CONNECTION RETCODE COMPILER
OPERATION MSGTYPE INDEX BUFFER.
```

Parameters

CONNECTION

(I) Handle for this connection. This connection handle must already be allocated with CTBCONALLOC.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under “Return value,” in this section.

COMPILER

This argument is ignored.

OPERATION

(I) Operation to perform. Assign this argument one of the following values:

Value	Meaning
CS-GET (33)	Retrieves a specific message
CS-CLEAR (35)	Clears message information for this connection.
CS-INIT (36)	Initializes in-line error handling.
CS-STATUS (37)	Returns the current number of stored messages.
CS-MSGLIMIT (38)	Sets the maximum number of messages to store.

MSGTYPE

(I) Type of message or structure on which the operation is to be performed. *MSGTYPE* can be any of the following symbolic values:

CS-CLIENTMSG-TYPE (4700)	A CLIENTMSG structure. Indicates Client-Library messages.
CS-SERVERMSG-TYPE (4701)	A SERVERMSG structure. Indicates messages sent by the Mainframe ClientConnect or other server.

CS-ALLMSG-TYPE (4702)	Operation is performed on both Client-Library and server messages.
SQLCA-TYPE (4703)	A SQLCA structure.
SQLCODE-TYPE (4704)	A SQLCODE structure.

INDEX

(I) Index number of the message being retrieved. Messages are numbered sequentially: the first message has an index of 1, the second an index of 2, and so forth.

- If *MSGTYP* is CS-CLIENTMSG-TYPE, then *INDEX* refers to Client-Library messages only.
- If *MSGTYP* is CS-SERVERMSG-TYPE, then *INDEX* refers to server messages only.
- If *MSGTYP* is CS-ALLMSG-TYPE, then *INDEX* refers to both Client-Library and server messages.
- *INDEX* should be initialized to 1.

BUFFER

(I/O) An integer or a variable ("buffer") that contains the message. Table 2-11 shows the relationship between *BUFFER* and other arguments.

This argument is typically either CHAR, a SQLCA structure, or a CLIENTMSG or SERVERMSG structure.

Note It is the responsibility of the programmer to provide a buffer large enough to hold the largest possible message. If the buffer is too small, the message will overwrite data in adjacent fields.

Return value

CTBDIAG returns one of the following values:

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.
CS-FAIL (-2)	The routine failed. Common reasons for a CTBDIAG failure include: • Invalid CONNECTION. • Inability to allocate memory. • Invalid parameter (for example, parameter is not allowed for operation). • Invalid parameter combination.

Value	Meaning
CS-NOERR (-207)	The application attempted to retrieve a message for which the index number is greater than the number of messages in the queue. For example, the application attempted to retrieve message number 3, when only 2 messages are queued.

Examples

Example 1

The following example uses CTBDIAG to prepare to receive messages. This example is taken from the sample program SYCTSAA5 in Appendix A, "Sample Language Requests."

```
 *============================================================
*==                                                        ==
*== Subroutine to process input data                       ==
*==                                                        ==
*============================================================
     PROCESS-INPUT.


    *********************************
    * ALLOCATE A CONNECTION HANDLE. *
    *********************************

         MOVE ZERO TO CSL-CON-HANDLE.

         CALL 'CTBCONAL' USING CSL-CTX-HANDLE
                               CSL-RC
                               CSL-CON-HANDLE.

         IF CSL-RC NOT EQUAL CS-SUCCEED
           THEN
              MOVE SPACES TO MSGSTR
              STRING 'CTBCONAL failed' DELIMITED BY SIZE INTO MSGSTR
              PERFORM PRINT-MSG
              PERFORM ALL-DONE
         END-IF.

    ******************
    * SET THE USER ID *
    ******************

         CALL 'CTBCONPR' USING CSL-CON-HANDLE
                               CSL-RC
                               CS-SET
                               CS-USERNAME
                               PF-USER
```

```
                              PF-USER-SIZE
                              CS-FALSE
                              OUTLEN.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBCONPR for user-id failed' DELIMITED BY SIZE
                                               INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
     END-IF.

*********************
* SET THE PASSWORD *
*********************

     CALL 'CTBCONPR' USING CSL-CON-HANDLE
                           CSL-RC
                           CS-SET
                           CS-PASSWORD
                           PF-PWD
                           PF-PWD-SIZE
                           CS-FALSE
                           OUTLEN.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBCONPR for password failed' DELIMITED BY SIZE
                                                INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
     END-IF.

*********************
* SET THE TRAN NAME *
*********************

     IF PF-TRAN-SIZE IS NOT EQUAL TO ZEROES THEN

         CALL 'CTBCONPR' USING CSL-CON-HANDLE
                               CSL-RC
                               CS-SET
                               CS-TRANSACTION-NAME
                               PF-TRAN
```

```
                                PF-TRAN-SIZE
                                CS-FALSE
                                OUTLEN

           IF CSL-RC NOT EQUAL CS-SUCCEED
             THEN
                MOVE SPACES TO MSGSTR
                STRING 'CTBCONPR for TRAN name failed'
                       DELIMITED BY SIZE INTO MSGSTR
                PERFORM PRINT-MSG
                PERFORM ALL-DONE
             END-IF

      END-IF.

*****************************
* SET THE NET DRIVER PROPERTY *
*****************************

      IF PF-NETDRV = SPACES OR PF-NETDRV = 'LU62'                X
                            OR PF-NETDRV = 'lu62'
          MOVE CS-LU62 TO NETDRIVER
      ELSE
        IF PF-NETDRV = 'IBMTCPIP' OR PF-NETDRV = 'ibmtcpip'
          MOVE CS-TCPIP TO NETDRIVER
      ELSE
        IF PF-NETDRV = 'INTERLIN' OR PF-NETDRV = 'interlin'
          MOVE CS-INTERLINK TO NETDRIVER
      ELSE
        IF PF-NETDRV = 'CPIC' OR PF-NETDRV = 'cpic'
          MOVE CS-NCPIC TO NETDRIVER
      END-IF.

      IF PF-DRV-SIZE IS NOT EQUAL TO ZEROES THEN

          CALL 'CTBCONPR' USING CSL-CON-HANDLE
                                CSL-RC
                                CS-SET
                                CS-NET-DRIVER
                                NETDRIVER
                                CS-UNUSED
                                CS-FALSE
                                OUTLEN

          IF CSL-RC NOT EQUAL CS-SUCCEED
            THEN
```

```
                MOVE SPACES TO MSGSTR
                STRING 'CTBCONPR for network driver failed'
                       DELIMITED BY SIZE INTO MSGSTR
                PERFORM PRINT-MSG
                PERFORM ALL-DONE
              END-IF

      END-IF.

*********************************
* SET FOR MAINFRAME EXTRA INFO *
*********************************

      CALL 'CTBCONPR' USING CSL-CON-HANDLE
                            CSL-RC
                            CS-SET
                            CS-EXTRA-INF
                            CS-TRUE
                            CS-UNUSED
                            CS-FALSE
                            CS-UNUSED.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBCONPR for extra info failed'
                              DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
      END-IF.

***********************************
* SETUP retrieval of All Messages *
***********************************

      CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                           CSL-RC,
                           CS-UNUSED,
                           CS-INIT,
                           CS-ALLMSG-TYPE,
                           CS-UNUSED,
                           CS-UNUSED.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
```

```
           STRING 'CTBDIAG CS-INIT failed' DELIMITED BY SIZE
                                                   INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
       END-IF.

******************************************
* set the upper limit of number of messages *
******************************************

       MOVE 5 TO PF-MSGLIMIT.

       CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                            CSL-RC,
                            CS-UNUSED,
                            CS-MSGLIMIT,
                            CS-ALLMSG-TYPE,
                            CS-UNUSED,
                            PF-MSGLIMIT.

       IF CSL-RC NOT EQUAL CS-SUCCEED
         THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBDIAG CS-MSGLIMIT failed' DELIMITED BY SIZE
                                                   INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
       END-IF.

************************
* CONNECT TO THE SERVER *
************************

       CALL 'CTBCONNE' USING CSL-CON-HANDLE
                             CSL-RC
                             PF-SERVER
                             PF-SERVER-SIZE
                             CS-FALSE.

       IF CSL-RC NOT EQUAL CS-SUCCEED
         THEN
            MOVE SPACES TO MSGSTR
            STRING 'CTBCONNE failed' DELIMITED BY SIZE INTO MSGSTR
            PERFORM PRINT-MSG
            PERFORM ALL-DONE
       END-IF.
```

```
     IF NO-ERRORS
       THEN
         PERFORM SEND-COMMAND
     END-IF.
```

Example 2

The following example uses CTBDIAG to retrieve diagnostic messages. This example is taken from the sample program SYCTSAA5 in Appendix A, "Sample Language Requests."

```
 *=====================================================
 *==                                                 ==
 *== Subroutine to retrieve any diagnostic messages ==
 *==                                                 ==
 *=====================================================
   GET-DIAG-MESSAGES.

**********************************
* Disable calls to this subroutine *
**********************************

     MOVE 'N' TO SW-DIAG.

*****************************
* First, get client messages *
*****************************

     CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                          CSL-RC,
                          CS-UNUSED,
                          CS-STATUS,
                          CS-CLIENTMSG-TYPE,
                          CS-UNUSED,
                          DG-NUM-OF-MSGS.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBDIAG CS-STATUS CS-CLIENTMSG-TYP fail'
                         DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
       ELSE
         IF DG-NUM-OF-MSGS > 0
           THEN
```

```
               PERFORM RETRIEVE-CLIENT-MSGS
                   VARYING I FROM 1 BY 1
                       UNTIL I IS GREATER THAN DG-NUM-OF-MSGS
           END-IF
       END-IF.

****************************
* Then, get server messages *
****************************

       CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                            CSL-RC,
                            CS-UNUSED,
                            CS-STATUS,
                            CS-SERVERMSG-TYPE,
                            CS-UNUSED,
                            DG-NUM-OF-MSGS.

       IF CSL-RC NOT EQUAL CS-SUCCEED
         THEN
           STRING 'CTBDIAG CS-STATUS CS-SERVERMSG-TYP fail'
                           DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
         ELSE
           IF DG-NUM-OF-MSGS > 0
             THEN
               PERFORM RETRIEVE-SERVER-MSGS
                   VARYING I FROM 1 BY 1
                       UNTIL I IS GREATER THAN DG-NUM-OF-MSGS
           END-IF
       END-IF.

 GET-DIAG-MESSAGES-EXIT.
       EXIT.

*=============================================================
*==                                                         ==
*== Subroutine to retrieve diagnostic messages from client ==
*==                                                         ==
*=============================================================
 RETRIEVE-CLIENT-MSGS.

       MOVE 1 TO I1.

       CALL 'CTBDIAG' USING CSL-CON-HANDLE,
```

```
                              CSL-RC,
                              CS-UNUSED,
                              CS-GET,
                              CS-CLIENTMSG-TYPE,
                              DG-MSGNO,
                              CLIENT-MSG.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBDIAG CS-GET CS-CLIENTMSG-TYPE failed'
                          DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.

************************
* display message text *
************************

      MOVE DISP-CLIENT-MSG-HDR TO RSLTNO( I1 ).
      MOVE 3 TO I1.

      MOVE CM-SEVERITY       TO CM-SEVERITY-DATA.
      MOVE CM-STATUS         TO CM-STATUS-DATA.
      MOVE DISP-CLIENT-MSG-1 TO RSLTNO( I1 ).
      COMPUTE I1 EQUAL I1 + 1

      MOVE CM-MSGNO          TO CM-OC-MSGNO-DATA.
      MOVE DISP-CLIENT-MSG-2 TO RSLTNO( I1 ).
      COMPUTE I1 EQUAL I1 + 1

      IF CM-MSGNO NOT EQUAL 0
        THEN
          MOVE SPACES            TO CM-OC-MSG-DATA
          MOVE CM-TEXT           TO CM-OC-MSG-DATA
          MOVE CM-TEXT           TO DISP-CLIENT-MSG-3A
          MOVE DISP-CLIENT-MSG-3 TO RSLTNO( I1 )
          COMPUTE I1 EQUAL I1 + 1
          IF CM-TEXT-LEN > 66
            THEN
              MOVE CM-OC-MSG-DATA-2   TO CM-OC-MSG-DATA-X
              MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
              COMPUTE I1 EQUAL I1 + 1
              IF CM-TEXT-LEN > 132
                THEN
```

```
              MOVE SPACES             TO CM-OC-MSG-DATA-X
              MOVE CM-OC-MSG-DATA-3   TO CM-OC-MSG-DATA-X
              MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
              COMPUTE I1 EQUAL I1 + 1
              IF CM-TEXT-LEN > 198
                THEN
                  MOVE SPACES             TO CM-OC-MSG-DATA-X
                  MOVE CM-OC-MSG-DATA-4   TO CM-OC-MSG-DATA-X
                  MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
                  COMPUTE I1 EQUAL I1 + 1
              END-IF
          END-IF
        END-IF
    ELSE
      MOVE DISP-EMPTY-CLIENT-MSG-3 TO RSLTNO( I1 )
      COMPUTE I1 EQUAL I1 + 1
END-IF.

MOVE CM-OS-MSGNO       TO CM-OS-MSGNO-DATA.
MOVE DISP-CLIENT-MSG-4 TO RSLTNO( I1 ).
COMPUTE I1 EQUAL I1 + 1

IF CM-OS-MSGNO NOT EQUAL 0
  THEN
    MOVE SPACES            TO CM-OS-MSG-DATA
    MOVE CM-OS-MSGTXT      TO CM-OS-MSG-DATA
    MOVE SPACES            TO DISP-CLIENT-MSG-5A
    MOVE CM-OS-MSGTXT      TO DISP-CLIENT-MSG-5A
    MOVE DISP-CLIENT-MSG-5 TO RSLTNO( I1 )
    COMPUTE I1 EQUAL I1 + 1
    IF CM-OS-MSGTEXT-LEN > 66
      THEN
        MOVE SPACES             TO CM-OC-MSG-DATA-X
        MOVE CM-OS-MSG-DATA-2   TO CM-OC-MSG-DATA-X
        MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
        COMPUTE I1 EQUAL I1 + 1
        IF CM-OS-MSGTEXT-LEN > 132
          THEN
            MOVE SPACES             TO CM-OC-MSG-DATA-X
            MOVE CM-OS-MSG-DATA-3   TO CM-OC-MSG-DATA-X
            MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
            COMPUTE I1 EQUAL I1 + 1
            IF CM-OS-MSGTEXT-LEN > 198
              THEN
                MOVE SPACES             TO CM-OC-MSG-DATA-X
                MOVE CM-OS-MSG-DATA-4   TO CM-OC-MSG-DATA-X
```

```
                    MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
                    COMPUTE I1 EQUAL I1 + 1
                END-IF
            END-IF
          END-IF
        ELSE
          MOVE DISP-EMPTY-CLIENT-MSG-5 TO RSLTNO( I1 )
          COMPUTE I1 EQUAL I1 + 1
      END-IF.

 RETRIEVE-CLIENT-MSGS-EXIT.
      EXIT.

*============================================================*
*==                                                        ==
*== Subroutine to retrieve diagnostic messages from server ==
*==                                                        ==
*============================================================*
 RETRIEVE-SERVER-MSGS.

      CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                           CSL-RC,
                           CS-UNUSED,
                           CS-GET,
                           CS-SERVERMSG-TYPE,
                           DG-MSGNO,
                           SERVER-MSG.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBDIAG CS-GET CS-SERVERMSG-TYPE failed'
                          DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.

************************
* display message text *
************************

      MOVE SM-MSGNO   TO SM-MSG-NO-DATA.
      MOVE SM-SEV     TO SM-SEVERITY-DATA.
      MOVE SM-STATE   TO SM-STATE-DATA.

      MOVE SM-LINE    TO SM-LINE-NO-DATA.
```

```
           MOVE SM-STATUS  TO SM-STATUS-DATA.

           MOVE SPACES     TO SM-SVRNAME-DATA.
           MOVE SM-SVRNAME TO SM-SVRNAME-DATA.

           MOVE SPACES     TO SM-PROC-ID-DATA.
           MOVE SM-PROC    TO SM-PROC-ID-DATA.

           MOVE SPACES     TO SM-MSG-DATA.
           MOVE SM-TEXT    TO SM-MSG-DATA.

           MOVE SPACES     TO DISP-SERVER-MSG-5A.
           MOVE SM-TEXT    TO DISP-SERVER-MSG-5A.


           MOVE DISP-SERVER-MSG-HDR TO RSLTNO (1).
           MOVE DISP-SERVER-MSG-1   TO RSLTNO (3).
           MOVE DISP-SERVER-MSG-2   TO RSLTNO (4).
           MOVE DISP-SERVER-MSG-3   TO RSLTNO (5).
           MOVE DISP-SERVER-MSG-4   TO RSLTNO (6).

           MOVE DISP-SERVER-MSG-5   TO RSLTNO (7).
           IF SM-TEXT-LEN > 66
             THEN
               MOVE SPACES             TO SM-MSG-DATA-X
               MOVE SM-MSG-DATA-2      TO SM-MSG-DATA-X
               MOVE DISP-SERVER-MSG-5X TO RSLTNO(8)
               IF SM-TEXT-LEN > 132
                 THEN
                   MOVE SPACES             TO SM-MSG-DATA-X
                   MOVE SM-MSG-DATA-3      TO SM-MSG-DATA-X
                   MOVE DISP-SERVER-MSG-5X TO RSLTNO(9)
                   IF SM-TEXT-LEN > 198
                     THEN
                       MOVE SPACES             TO SM-MSG-DATA-X
                       MOVE SM-MSG-DATA-4      TO SM-MSG-DATA-X
                       MOVE DISP-SERVER-MSG-5X TO RSLTNO(10)
                   END-IF
               END-IF
           END-IF.

       RETRIEVE-SERVER-MSGS-EXIT.
           EXIT.
```

Usage

- CTBDIAG manages in-line message handling for a specific connection. If an application has more than one connection, it must make separate CTBDIAG calls for each connection.

- Open ClientConnect applications always use CTBDIAG to handle Client-Library and server messages. Applications built with Open Client can provide alternative message-handling facilities.
- An application can perform operations on Client-Library messages, server messages, or both.

  For example, an application can clear Client-Library messages without affecting server messages:

  ```
  CALL 'CTBDIAG' USING CONNECTION RETCODE CS-UNUSED
  CS-CLEAR CS-CLIENTMSG CS-UNUSED MSGBUFFER.
  ```

- CTBDIAG allows an application to retrieve message information into standard Client-Library structures (CLIENTMSG, SERVERMSG, SQLCA or SQLCODE).

  When retrieving messages, CTBDIAG assumes that *BUFFER* points to a structure of the type indicated by *MSGTYPE*.

  - An application that is retrieving messages into a SQLCA or SQLCODE structure must set the Client-Library property CS-EXTRA-INF to CS-TRUE. This is because the SQL structures require information that is not ordinarily returned by the Client-Library error handling mechanism.

    Use CTBCONPROPS or CSBCONFIG to set CS-EXTRA-INF.
  - An application that does not use the SQLCA or SQLCODE structures can also set CS-EXTRA-INF to CS-TRUE. In this case, the extra information returns as standard Client-Library messages.

    For more information about CS-EXTRA-INF, see “Properties” on page 181 and the reference pages for CTBCONPROPS and CSBCONFIG.

Warning! If CTBDIAG does not have sufficient internal storage space in which to save a new message, it throws away all unread messages and stops saving messages. The next time it is called with *OPERATION* as CS-GET, it returns a message to indicate the space problem. After returning this message, CTBDIAG starts saving messages again.

Initializing in-line error handling

- An application must initialize in-line error handling before it can retrieve any errors. To initialize in-line error handling, call CTBDIAG with *OPERATION* as CS-INIT.

- Generally, if a connection uses in-line error handling, the application should call CTBDIAG to initialize in-line error handling for a connection immediately after allocating it with CTBCONALLOC.

Clearing messages

- To clear message information for a connection, an application calls *OPERATION* as CS-CLEAR.
    - To clear Client-Library messages only, set *MSGTYPE* to CS-CLIENTMSG-TYPE.
    - To clear server messages only, set *MSGTYPE* to CS-SERVERMSG-TYPE.
    - To clear both Client-Library and server messages, set *MSGTYPE* to SQLCA, SQLCODE, or CS-ALLMSG-TYPE.
- If *OPERATION* is CS-CLEAR and *MSGTYPE* is not CS-ALLMSG-TYPE:
    - CTBDIAG assumes that *BUFFER* is a structure of type *MSGTYPE*.
    - CTBDIAG clears the buffer by setting it to blanks or LOW-VALUES, as appropriate.
- Message information is not cleared until an application explicitly calls CTBDIAG with *OPERATION* as CS-CLEAR. Retrieving a message does not remove it from the message queue.

Retrieving messages

- To retrieve message information, an application calls CTBDIAG with *OPERATION* as CS-GET, *MSGTYPE* as the type of structure in which to retrieve the message, *INDEX* as the index number of the message of interest, and *BUFFER* as an integer or a variable, as appropriate.
- If *MSGTYPE* is CS-CLIENTMSG-TYPE, *INDEX* refers only to Client-Library messages.
- If *MSGTYPE* is CS-SERVERMSG-TYPE, *INDEX* refers only to server messages.
- If *MSGTYPE* has any other value, *INDEX* refers to the collective "queue" of both types of messages combined.
- CTBDIAG creates a messages queue in the buffer and fills the buffer with message information. It returns messages to the client in the order in which they are received.

- If an application attempts to retrieve a message with an index that is higher than the highest valid index, CTBDIAG returns CS-NOMSG to indicate that a message is not available.

Limiting messages

- The Client-Library default behavior is to save an unlimited number of messages. Applications running on platforms with limited memory may want to limit the number of messages that Client-Library saves. The default for MVS is 25.

  An application can limit the number of saved Client-Library messages, the number of saved server messages, and the total number of saved messages.

- To limit the number of saved messages, an application calls CTBDIAG with *OPERATION* as CS-MSGLIMIT and *MSGTYPE* as CS-CLIENTMSG-TYPE, CS-SERVERMSG-TYPE, or CS-ALLMSG-TYPE:

  - If *MSGTYPE* is CS-CLIENTMSG-TYPE, the number of Client-Library messages is limited.
  - If *MSGTYPE* is CS-SERVERMSG-TYPE, the number of server messages is limited.
  - If *MSGTYPE* is CS-ALLMSG-TYPE, the total number of Client-Library and server messages combined is limited.

- When a specific message limit is reached, Client-Library discards any new messages of that type. When a combined message limit is reached, Client-Library discards any new messages.

Retrieving the number of messages

- To find out how many messages were retrieved, an application calls CTBDIAG with *OPERATION* as CS-STATUS and *MSGTYPE* as the type of message of interest.

  Table 2-11 on page 102 lists a summary of arguments for CTBDIAG.

Table 2-11: Summary of arguments (CTBDIAG)

Operation	CTBDIAG action	MSGTYPE value	INDEX value	BUFFER value
CS-INIT	Initializes in-line error handling. An application must call CTBDIAG with a CS-INIT operation before it can process error messages in line.	CS-UNUSED	CS-UNUSED	Ignored.
CS-CLEAR	Clears message information for this connection.	One of the legal MSGTYPE values. • If *MSGTYPE* is CS-CLIENTMSG-TYPE, CTBDIAG clears Client-Library messages only. • If *MSGTYPE* is CS-SERVERMSG-TYPE, CTBDIAG clears server messages only. • If *BUFFER* is not LOW-VALUES and *MSGTYPE* is not CS-ALLMSG-TYPE, then CTBDIAG clears the buffer by initializing it with blanks and/or zeroes.	CS-UNUSED	A buffer for which the type is defined by *MSGTYPE* or is LOW-VALUES.

Operation	CTBDIAG action	MSGTYPE value	INDEX value	BUFFER value
CS-GET	Retrieves a specific message.	Any legal *MSGTYPE* value except CS-ALLMSG-TYPE. • If *MSGTYPE* is CS-CLIENTMSG-TYPE,CTBDIAG retrieves a Client-Library message into a CLIENTMSG structure. • If *MSGTYPE* is CS-SERVERMSG-TYPE, CTBDIAG retrieves a server message into a SERVERMSG structure. • If *MSGTYPE* has any other value, CTBDIAG retrieves either a server message or a Client-Library message.	The index number of the message to retrieve.	A buffer whose type is defined by *MSGTYPE*.
CS-MSGLIMIT	Sets the maximum number of messages to store.	CS-CLIENTMSG-TYPE to limit Client-Library messages only. CS-SERVERMSG-TYPE to limit server messages only. CS-ALLMSG-TYPE to limit the total number of Client-Library and server messages combined.	CS-UNUSED	An integer value.

Operation	CTBDIAG action	MSGTYPE value	INDEX value	BUFFER value
CS-STATUS	Returns the current number of stored messages.	CS-CLIENTMSG-TYPE to retrieve the number of Client-Library messages. CS-SERVERMSG-TYPE to retrieve the number of server messages. CS-ALLMSG-TYPE to retrieve the total number of Client-Library and server messages combined.	CS-UNUSED	An integer variable.

See also

Related topics

- "CLIENTMSG structure" on page 168
- "Error and message handling" on page 177
- "SERVERMSG structure" on page 194
- "SQLCA structure" on page 197

CTBEXIT

Description

Exits Client-Library.

Syntax

```
COPY CTPUBLIC.

01 CONTEXT        PIC S9(9) COMP SYNC.
01 RETCODE        PIC S9(9) COMP SYNC.
01 OPTION         PIC S9(9) COMP SYNC.

CALL 'CTBEXIT' USING CONTEXT RETCODE OPTION.
```

Parameters

CONTEXT

(I) A context structure. The context structure is defined in the program call CSBCTXALLOC. This value identifies the Client-Library context being exited.

If this value is invalid or nonexistent, CTBEXIT fails.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under "Return value," in this section.

OPTION

(O) Indicator specifying whether or not CTBEXIT closes connections for which results are pending.

CTBEXIT behaves in differently, depending on the value specified for *OPTION*. The following table lists the symbolic values that are legal for *OPTION*:

OPTION Value	CTBEXIT Action
CS-UNUSED (-99999)	Closes all open connections for which no results are pending and terminates Client-Library for this context. If results are pending for one or more connections, CTBEXIT returns CS-FAIL and does not terminate Client-Library.
CS-FORCE-EXIT (300)	Closes all open connections for this context, whether or not any results are pending, and terminates Client-Library for this context.

Return value

CTBEXIT returns one of the following values:

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.
CS-FAIL (-2)	The routine failed.
TDS-INVALID-PARAMETER (-4)	A parameter contains an illegal value.
TDS-RESULTS-STILL-ACTIVE (-50)	Some results are still pending.
TDS-CONNECTION-TERMINATED (-4997)	The connection is not active.
TDS-WRONG-STATE (-6)	Program is in the wrong communication state to issue this call.

Examples

The following code fragment demonstrates the use of CTBEXIT at the end of a program, after results processed. It is taken from the sample program SYCTSAA5 in Appendix A, "Sample Language Requests."

```
 *================================================
*==                                              ==
*== Subroutine to perform exit client library and ==
*== deallocate context structure.                ==
*==                                              ==
*================================================
    QUIT-CLIENT-LIBRARY.

   ***************************
   * EXIT THE CLIENT LIBRARY *
```

```
***************************
     CALL 'CTBEXIT' USING CSL-CTX-HANDLE
                          CSL-RC
                          CS-UNUSED.
     IF CSL-RC = CS-FAIL
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBEXIT failed' DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
     END-IF.
***********************************
* DE-ALLOCATE THE CONTEXT STRUCTURE *
***********************************
     CALL 'CSBCTXDR' USING CSL-CTX-HANDLE
                           CSL-RC.
     IF CSL-RC = CS-FAIL
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CSBCTXDR failed' DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
     END-IF.
     EXEC CICS RETURN END-EXEC.
 QUIT-CLIENT-LIBRARY-EXIT.
     EXIT.
```

Usage

- CTBEXIT terminates a Client-Library context. It closes all open connections, deallocates internal data space and cleans up any platform-specific initialization.
- CTBEXIT must be the last Client-Library routine called within a Client-Library context.
- If an application needs to call Client-Library routines after it calls CTBEXIT, it can re-initialize Client-Library by calling CTBINIT again.
- If results are pending on any of the context connections and *OPTION* is not passed as CS-FORCE-EXIT, CTBEXIT returns CS-FAIL. This means that Client-Library is not correctly terminated. The application must handle the pending results before calling CTBEXIT, or it can call CTBEXIT again, specifying CS-FORCE-EXIT.
- To close a single connection, an application calls CTBCLOSE.
- If CTBINIT is called for a context, the application must call CTBEXIT before it calls CSBCTXDROP to deallocate the context.

See also

Related functions

- CTBCLOSE on page 37

• CTBINIT on page 117

CTBFETCH

Description

Fetches result data.

Syntax

```
COPY CTPUBLIC.

01 COMMAND          PIC S9(9) COMP SYNC.
01 RETCODE          PIC S9(9) COMP SYNC.
01 TYP              PIC S9(9) COMP SYNC.
01 OFFSET           PIC S9(9) COMP SYNC.
01 OPTION           PIC S9(9) COMP SYNC.
01 ROWS-READ        PIC S9(9) COMP SYNC.

CALL 'CTBFETCH' USING COMMAND RETCODE TYP OFFSET OPTION
ROWS-READ.
```

Parameters

COMMAND

(I) Handle for this client/server operation. This handle is defined in the associated CTBCMDALLOC call.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under “Return value,” in this section.

TYP

(I) This argument is currently unused and should be passed as CS-UNUSED in order to ensure compatibility with future versions of Client-Library.

OFFSET

(I) This argument is currently unused and should be passed as CS-UNUSED in order to ensure compatibility with future versions of Client-Library.

OPTION

(I) This argument is currently unused and should be passed as CS-UNUSED in order to ensure compatibility with future versions of Client-Library.

ROWS-READ

(I) Variable where the number of result rows is returned. This variable is of type integer. CTBFETCH sets *ROWS-READ* to the number of rows read by the CTBFETCH call. This argument is required.

Return value

CTBFETCH returns one of the following values listed in Table 2-12.

Table 2-12: CTBFETCH return values

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully. CTBFETCH places the total number of rows read in ROWS-READ.
CS-FAIL (-2)	The routine failed. CTBFETCH places the number of rows fetched before the failure occurred in *ROWS-READ*. A common reason for a CTBFETCH failure is that a program variable specified through CTBBIND is too small to hold a fetched data item.
CS-CANCELLED (-202)	The operation was canceled. CTBFETCH places the number of rows fetched before the cancel occurred in *ROWS-READ*.
CS-ROW-FAIL (-203)	A recoverable error occurred while fetching a row. Recoverable errors include memory allocation failures and conversion errors that occur while copying row values to program variables. An application can continue calling CTBFETCH to continue retrieving rows, or can call CTBCANCEL to cancel the remaining results. CTBFETCH places the number of rows fetched before the error occurred in *ROWS-READ*, then continues by fetching the row after the error.
CS-END-DATA (-204)	No more rows are available in this result set. (Note that this is also a successful completion.
TDS-INVALID-PARAMETER (-4)	One of the CTBFETCH arguments contains an illegal value. The most likely cause of this code is assigning a value other than CS-UNUSED to one of more of the reserved arguments, *TYP*, *OFFSET*, and *OPTION*.
TDS-WRONG-STATE (-6)	Program is in the wrong communication state to issue this call. It is in Send state instead of Receive state.

Examples

The following example shows a typical use of CTBFETCH. It is taken from the sample program SYCTSAA5 in Appendix A, “Sample Language Requests.”

```
 *=========================================================
 *==                                                     ==
 *== Subroutine to fetch row processing                  ==
 *==                                                     ==
 *=========================================================
FETCH-ROW-PROCESSING.
        CALL 'CTBFETCH' USING CSL-CMD-HANDLE,
                              CSL-RC,
                              CS-UNUSED,
                              CS-UNUSED,
                              CS-UNUSED,
                              FF-ROWS-READ.
```

```
      EVALUATE CSL-RC
          WHEN CS-SUCCEED
               MOVE 'Y'              TO SW-FETCH
               MOVE CS-VARCHAR-TYPE TO DF-DATATYPE
               MOVE LENGTH OF CF-COL-FIRSTNME-TXT
                                     TO DF-MAXLENGTH
               MOVE CS-CHAR-TYPE     TO DF2-DATATYPE
               MOVE LENGTH OF CF-COL-FIRSTNME-CHAR
                                     TO DF2-MAXLENGTH
               CALL 'CSBCONVE' USING CSL-CTX-HANDLE,
                                      CSL-RC,
                                      DATAFMT,
                                      CF-COL-FIRSTNME,
                                      DATAFMT2,
                                      CF-COL-FIRSTNME-CHAR,
                                      CF-COL-LEN
               IF CSL-RC NOT EQUAL CS-SUCCEED
                 THEN
                    MOVE SPACES TO MSGSTR
                    STRING 'CSBCONVERT CS-VARCHAR-TYPE failed'
                                DELIMITED BY SIZE INTO MSGSTR
                    PERFORM PRINT-MSG
                    PERFORM ALL-DONE
               END-IF
               COMPUTE FF-ROW-NUM = FF-ROW-NUM + 1
*************************************
* save ROW RESULTS for later display *
*************************************
               MOVE CF-COL-FIRSTNME-CHAR TO
                    OR-COL-FIRSTNME-CHAR
               MOVE DATA-SMALLINT TO
                    OR-COL-EDUCLVL
               IF FF-ROW-NUM > MAX-SCREEN-ROWS
                 THEN
                   STRING 'Please press return to continue.'
                          DELIMITED BY SIZE INTO MSG1O
                   MOVE SPACES TO MSG-TEXT-2
                   PERFORM DISP-DATA
                   PERFORM CLEAR-SCREEN-DATA
                           VARYING FF-ROW-NUM FROM 1 BY 1
                           UNTIL FF-ROW-NUM > MAX-SCREEN-ROWS
                   MOVE LOW-VALUES TO A5PANELO
                   COMPUTE PAGE-CNT = PAGE-CNT + 1
                   MOVE 1 TO FF-ROW-NUM
**-------------------------------------------------------------
**   Setup column headings
```

```
**-------------------------------------------------------------
                MOVE 'FirstName    EducLvl' TO
                                RSLTNO(FF-ROW-NUM)
                COMPUTE FF-ROW-NUM = FF-ROW-NUM + 1
                MOVE '===========  =======' TO
                                RSLTNO(FF-ROW-NUM)
                COMPUTE FF-ROW-NUM = FF-ROW-NUM + 1
            END-IF
            MOVE OUTPUT-ROW-STR TO RSLTNO(FF-ROW-NUM)
            MOVE SPACES         TO CF-COL-FIRSTNME-TXT
        WHEN CS-END-DATA
            MOVE SPACES     TO MSG1O
            MOVE 'N'        TO SW-FETCH
            MOVE 'Press Clear To Exit'
                            TO MSG-TEXT-2
            STRING 'All rows processing completed!'
                   DELIMITED BY SIZE INTO MSG1O
            PERFORM DISP-DATA
        WHEN CS-FAIL
            MOVE 'N'    TO SW-FETCH
            MOVE SPACES TO MSGSTR
            STRING 'CTBFETCH returned CS-FAIL ret-code'
                   DELIMITED BY SIZE INTO MSGSTR
            PERFORM PRINT-MSG
        WHEN CS-ROW-FAIL
            MOVE 'N'    TO SW-FETCH
            MOVE SPACES TO MSGSTR
            STRING 'CTBFETCH returned CS-ROW-FAIL ret-code'
                            DELIMITED BY SIZE INTO MSGSTR
            PERFORM PRINT-MSG
        WHEN CS-CANCELLED
            MOVE 'N'         TO SW-FETCH
            MOVE MF-CANCELED TO MSG1O
            PERFORM PRINT-MSG
        WHEN OTHER
            MOVE 'N'    TO SW-FETCH
            MOVE SPACES TO MSGSTR
            STRING 'CTBFETCH returned UNKNOWN ret-code'
                            DELIMITED BY SIZE INTO MSGSTR
            PERFORM PRINT-MSG
    END-EVALUATE.
 FETCH-ROW-PROCESSING-EXIT.
    EXIT.
```

Usage

- CTBFETCH fetches result data. “Result data” is an umbrella term for the various types of data that a server can return to an application. These types of data include:
  - Regular rows.
  - Return parameters, including both message parameters and RPC return parameters.
  - Stored procedure status results.

  CTBFETCH is used to fetch all of these types of data.
- Conceptually, result data is returned to an application in the form of one or more rows that make up a “result set.”

  Regular row result sets can contain more than one row. For example, a regular row result set might contain a hundred rows. If array binding is specified for the data items in a regular row result set, then multiple rows can be fetched with a single call to CTBFETCH. The number of rows fetched are returned in the *ROWS-READ* argument.

  Return parameters and status results, however, only contain a single row. For this reason, even if array binding is specified, only a single row of data is fetched.
- CTBRESULTS specifies the type of result available in the *RESULT-TYP* variable. CTBRESULTS must indicate a result type of CS-ROW-RESULT, CS-PARAM-RESULT, or CS-STATUS-RESULT before an application calls CTBFETCH.
- After calling CTBRESULTS, an application can do one of the following:
  - Process the result set by binding the result items and fetching the data, using CTBFETCH (optionally preceded by CTBDESCRIBE and CTBBIND).
  - Discard the result set, using CTBCANCEL.
- If an application does not cancel a result set, it must completely process the result set by repeatedly calling CTBFETCH as long as CTBFETCH continues to indicate that rows are available.

  The simplest way to do this is in a loop that terminates when CTBFETCH fails to return either CS-SUCCEED or CS-ROW-FAIL. After the loop terminates, an application can check the CTBFETCH final return code to find out what caused the termination.

Fetching regular rows

Regular rows can be fetched from the server one row at a time, or several rows at once.

- When fetching multiple rows, the number of rows to be fetched is indicated by the FMT-COUNT field in the DATAFMT structures used to bind the data items in the result set. Note that the FMT-COUNT field must have the same value for all CTBBIND calls for a result set.

  If FMT-COUNT is 0 or 1, CTBFETCH fetches one row.

Fetching return parameters

A return parameter result set contains either stored procedure return parameters or message parameters.

- A return parameter result set consists of a single row with a number of columns equal to the number of return parameters.

Fetching a return status

A stored procedure return status result set consists of a single row with a single column, containing the status.

See also

Related functions

- CTBBIND on page 25
- CTBDESCRIBE on page 78
- CTBRESULTS on page 137

Related documentation

- Mainframe Connect Client Option and Server Option *Messages and Codes*

CTBGETFORMAT

Description

Returns the user-defined format for a result column.

Note This function is used with requests to Adaptive Server only.

Syntax

```
COPY CTPUBLIC.

01 COMMAND PIC S9(9) COMP SYNC.
01 RETCODE PIC S9(9) COMP SYNC.
```

```
01 COLUMN-NUM PIC S9(9) COMP SYNC.
01 BUFFER PIC S9(9) or PIC X(9)
01 BUFFER-LEN PIC S9(9) COMP SYNC.
01 OUTLEN PIC S9(9) COMP SYNC.

CALL 'CTBGETFO' USING COMMAND RETCODE COLUMN-NUM BUFFER
BUFFER-LEN OUTLEN.
```

Parameters

COMMAND

(I) Handle for this client/server operation. This handle is defined in the associated CTBCMDALLOC call.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under "Return value," in this section.

COLUMN-NUM

(I) Number of the column for which the user-specified format is desired.

COLUMN-NUM refers to the select-list ID of the column. The first column in the select list of a select statement is column number 1, the second is column number 2, and so forth.

BUFFER

(O) Variable ("buffer") in which CTBGETFORMAT places the requested information.

This argument is typically:

```
01 BUFFER PIC X(n).
```

BUFFER-LEN

(I) Length, in bytes, of the buffer.

If *BUFFER-LEN* is too small to hold the requested information, CTBGETFORMAT sets *OUTLEN* to the length of the requested information, and returns CS-FAIL.

OUTLEN

(O) Length, in bytes, of the format string. *OUTLEN* is an integer variable where CTBGETFORMAT returns the total number of bytes being retrieved.

When the format string is larger than *BUFFER-LEN* bytes, an application uses this value to determine how many bytes are needed to hold the string.

If a format string is not associated with the specified column, CTBGETFORMAT sets *OUTLEN* to 0.

Return value

CTBGETFORMAT returns one of the following values:

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.
CS-FAIL (-2)	The routine failed.
TDS-INVALID-PARAMETER (-4)	One of the CTBGETFORMAT arguments contains an illegal value.

Examples

The following code fragment demonstrates the use of CTBGETFORMAT. This sample is not part of any sample program, so the working storage section is included here.

```
      01 CTX                       PIC S9(9) COMP SYNC VALUE +0.
      01 CON                       PIC S9(9) COMP SYNC VALUE +0.
      01 CMD                       PIC S9(9) COMP SYNC VALUE +0.
      01 RET                       PIC S9(9) COMP SYNC VALUE +0.
   01 RETCODE                   PIC S9(9) COMP SYNC VALUE +0.
   01 RESTYPE                   PIC S9(9) COMP SYNC VALUE +0.
   01 DATALEN                   PIC S9(9) COMP SYNC VALUE +0.
   01 COL-NUM                   PIC S9(9) COMP SYNC VALUE IS 0.
   01 FMT-BUFFER                PIC X(6).
   01 FMT-LEN                   PIC S9(9) COMP SYNC VALUE IS 0.
   01  SW-FETCH                 PIC X(01).
       88  NO-MORE-ROWS VALUE 'N'.
       88  MORE-ROWS    VALUE 'Y'.

   01 STRLENPIC S9(9) COMP SYNC.
   01 OUTLENPIC S9(9) COMP SYNC.
   01 USER-DATAPIC X(30).
   01 USER-BUFPIC X(8).
   01 NUMROWSPIC S9(9) COMP SYNC.

   PROCEDURE DIVISION.
   P0.
    .
    .
    .
RESULTS-PROCESSING.

* SET UP THE RESULTS DATA

     CALL 'CTBRESUL' USING CMD RETCODE RESTYPE.

* DETERMINE THE OUTCOME OF THE COMMAND EXECUTION

     EVALUATE RETCODE
```

```
           WHEN CS-SUCCEED

    * DETERMINE THE TYPE OF RESULT RETURNED BY THE CURRENT REQUEST
              EVALUATE RESTYPE

    * PROCESS ROW RESULTS
              WHEN CS-ROW-RESULT
                  PERFORM RESULT-GETFMT
                  PERFORM RESULT-CANCEL
                  PERFORM RESULT-ROW-PROCESSING
                  MOVE 'Y' TO SW-FETCH
                  PERFORM FETCH-ROW-PROCESSING UNTIL NO-MORE-ROWS
    RESULT-ROW-PROCESSING.

    * FOR EACH COLUMN, BIND THE RESULT
         PERFORM BIND-ROW-PROCESSING.

    RESULT-GETFMT.
       MOVE 1 TO COL-NUM.
       MOVE LENGTH OF FMT-BUFFER TO FMT-LEN.
       CALL 'CTBGETFO' USING CMD RETCODE COL-NUM FMT-BUFFER FMT-LEN
              FMT-LEN OUTLEN.
       IF RETCODE NOT EQUAL CS-SUCCEED
           STRING 'CTBGETFO FAILED' DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG.

    RESULT-CANCEL.
       CALL 'CTBCANCE' USING CON RETCODE CMD CS-CANCEL-ALL
              CS-UNUSED NUMROWS.
       IF RETCODE NOT EQUAL CS-SUCCEED
           STRING 'CTBCANCEL FAILED' DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG.
```

Usage

- CTBGETFORMAT returns the user-defined format, if any, for a result column. It indicates how the field should be formatted on screen.
- An application can call CTBGETFORMAT after CTBRESULTS indicates results of type CS-ROW-RESULT.
- For a description of how to add user-defined formats to Adaptive Server databases or Open Servers, see the Mainframe Connect Server Option *Installation and Administration Guide*.

See also

Related functions

- CTBBIND on page 25

- CTBDESCRIBE on page 78

CTBINIT

Description

Initializes Client-Library.

Syntax

```
COPY CTPUBLIC.

01 CONTEXT PIC S9(9) COMP SYNC.
01 RETCODE PIC S9(9) COMP SYNC.
01 VERSION  PIC S9(9) COMP SYNC.

CALL 'CTBINIT' USING CONTEXT RETCODE VERSION.
```

Parameters

CONTEXT

(I) A context structure. The context structure is defined in the program call CSBCTXALLOC. If this value is invalid or nonexistent, CTBINIT fails.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under "Return value," in this section.

VERSION

(I) Version of Client-Library behavior that the application expects.

The following table lists the symbolic values that are legal for *VERSION*:

Value	Meaning	Supported features
CS-VERSION-46	Application communicates with a version 4.6 Adaptive Server.	RPCs.
CS-VERSION-50	Application communicates with a version 10.0 Adaptive Server and above.	RPCs.

Return value

CTBINIT returns one of the following values:

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.
TDS-WRONG-STATE (-6)	Program is in the wrong communication state to issue this call. The most likely cause is that this context already initiated.

Examples

The following code fragment demonstrates the use of CTBINIT. It is taken from the sample program SYCTSAA5 in Appendix A, "Sample Language Requests."

```
*****************************
*    PROGRAM INITIALIZATION *
*****************************
       MOVE C-N    TO NO-MORE-MSGS-SW.
```

```
     MOVE C-N    TO NO-ERRORS-SW.
     MOVE C-Y    TO SW-DIAG.
     COMPUTE PAGE-CNT = PAGE-CNT + 1.
     PERFORM GET-SYSTEM-TIME.
     MOVE LOW-VALUES TO A5PANELO.
     MOVE -1         TO SERVERL.
 GET-INPUT-AGAIN.
     PERFORM DISPLAY-INITIAL-SCREEN.
     PERFORM GET-INPUT-DATA.
*********************************
*    ALLOCATE A CONTEXT STRUCTURE *
*********************************
     MOVE ZERO TO CSL-CTX-HANDLE.
     CALL 'CSBCTXAL' USING CS-VERSION-50
                           CSL-RC
                           CSL-CTX-HANDLE.
     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
          MOVE SPACES TO MSGSTR
          STRING 'CSBCTXAL failed' DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
     END-IF.
*********************************
* INTITIALIZE THE CLIENT-LIBRARY *
*********************************
     CALL 'CTBINIT' USING CSL-CTX-HANDLE
                          CSL-RC
                          CS-VERSION-50.
     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBINIT failed' DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
     END-IF.
     PERFORM PROCESS-INPUT.
     PERFORM QUIT-CLIENT-LIBRARY.
     GOBACK.
```

Usage

- CTBINIT initializes Client-Library. It sets up internal control structures and defines the version of Client-Library behavior that an application expects. Client-Library provides the requested behavior, regardless of the actual version of Client-Library in use.

- CTBINIT must be the first Client-Library routine call after CSBCTXALLOC. Other Client-Library routines fail if they are called before CTBINIT.
- Because an application calls CTBINIT before it sets up error handling, an application must check the CTBINIT return code to detect failure.
- It is not an error for an application to call CTBINIT multiple times. Some applications cannot guarantee which of several modules executes first. In such a case, each module should contain a call to CTBINIT.

See also

Related functions

- CSBCTXALLOC on page 160
- CTBEXIT on page 104

CTBPARAM

Description

Defines a command parameter.

Syntax

```
COPY CTPUBLIC.
01 COMMAND    PIC S9(9) COMP SYNC.
01 RETCODE    PIC S9(9) COMP SYNC.
01 DATAFMT
   05 FMT-NAME       PIC X(132).
   05 FMT-NAMELEN    PIC S9(9) COMP SYNC.
   05 FMT-TYPE       PIC S9(9) COMP SYNC.
   05 FMT-FORMAT     PIC S9(9) COMP SYNC.
   05 FMT-MAXLEN     PIC S9(9) COMP SYNC.
   05 FMT-SCALE      PIC S9(9) COMP SYNC.
   05 FMT-PRECIS     PIC S9(9) COMP SYNC.
   05 FMT-STATUS     PIC S9(9) COMP SYNC.
   05 FMT-COUNT      PIC S9(9) COMP SYNC.
   05 FMT-UTYPE      PIC S9(9) COMP SYNC.
   05 FMT-LOCALE     PIC S9(9) COMP SYNC.
01 DATA         type
01 DATALEN      PIC S9(9) COMP SYNC.
01 INDICATOR    PIC S9(4) COMP SYNC.

CALL 'CTBPARAM' USING COMMAND RETCODE DATAFMT DATA
DATALEN INDICATOR.
```

Parameters

COMMAND

(I) Handle for this client/server operation. This handle is defined in the associated CTBCMDALLOC call.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under “Return value,” in this section.

DATAFMT

(I) A structure that contains a description of the parameter. This structure is also used by CTBBIND, CTBDESCRIBE, and CSBCONVERT and is explained in “DATAFMT structure” on page 170.

Table 2-13 lists the fields in the DATAFMT structure, indicates whether or when they are used by CTBPARAM, and contains general information about the fields.

For specific information on how to set these fields when defining a parameter for a particular kind of command, see the charts in Table 2-13.

Note The programmer is responsible for adhering to these rules. Client-Library does not enforce them.

Table 2-13: Fields in the DATAFMT structure for CTBPARAM

When this field	Is used in this condition	Set the field to
FMT-NAME	When defining parameters for all supported commands.	The name of the parameter being defined. If FMT-NAMELEN is 0, the parameter is considered to be unnamed. Unnamed parameters are interpreted positionally. It is an error to mix named and unnamed parameters in a single command. **Note** When sending parameters to an Adaptive Server, FMT-NAME must begin with the "@" symbol, which prefixes all Adaptive Server stored procedure parameter names. When sending parameters with language requests, this must be the variable name as it appears in the language string. Transact-SQL names begin with the colon (:) symbol.
FMT-NAMELEN	When defining parameters for all supported commands.	The length, in bytes, of FMT-NAME. If FMT-NAMELEN is 0, the parameter is considered to be unnamed.
FMT-TYPE	When defining parameters for all supported commands.	The datatype of the parameter value. All datatypes listed under "Datatypes" on page 174 are valid.
FMT-FORMAT	Not used (CS-FMT-UNUSED).	Not applicable.
FMT-MAXLEN	When defining non-fixed-length return parameters for RPCs; otherwise CS-UNUSED.	The maximum length, in bytes, of the data returned in this parameter. For character or binary data, FMT-MAXLEN must represent the total length of the return parameter, including any space required for special terminating bytes, with this exception: when the parameter is a VARYCHAR datatype such as the DB2 VARCHAR, FMT-MAXLEN does not include the length of the "LL" length specification. For Sybase-decimal and Sybase-numeric, set FMT-MAXLEN to 35. If the parameter is non-return, if FMT-TYPE is fixed-length, or if the application does not need to restrict the length of return parameters, set FMT-MAXLEN to CS-UNUSED.
FMT-SCALE	Used for packed decimal, Sybase-decimal, and Sybase-numeric datatypes.	The number of digits after the decimal point.

When this field	Is used in this condition	Set the field to
FMT-PRECIS	Used for packed decimal, Sybase-decimal, and Sybase-numeric datatypes.	The total number of digits before and after the decimal point.
FMT-STATUS	When defining parameters for all types of commands except message commands.	The type of parameter being defined. One of the following values: • CS-INPUTVALUE - The parameter is an input parameter value for a non-return RPC parameter or a language request parameter. • CS-RETURN - The parameter is a return parameter.
FMT-COUNT	Not used (CS-FMT-UNUSED).	Not applicable.
FMT-UTYPE	Only when defining a parameter that has an Adaptive Server user-defined datatype; otherwise CS-UNUSED.	The user-defined datatype of the parameter, if any. FMT-UTYPE is set in addition to (not instead of) DATATYPE. **Note** This field is used for datatypes defined at the server, not for Open Client user-defined datatypes.
FMT-LOCALE	Not used (CS-FMT-UNUSED).	LOW-VALUES.

DATA
Variable that contains the parameter data.

To indicate a parameter value of LOW-VALUES, assign *INDICATOR* a value of -1.

If *INDICATOR* is -1, *DATA* and *DATA-LEN* are ignored. For example, an application might pass null parameters (containing LOW-VALUES) to a stored procedure or transaction that assigns default values to null input parameters.

DATA-LEN
The length, in bytes, of the parameter data. For Sybase-numeric and Sybase-decimal, set *DATA-LEN* to 35.

INDICATOR
An integer variable used to indicate a parameter value of LOW-VALUES. To indicate that a parameter is null, assign *INDICATOR* a value of -1. If *INDICATOR* is -1, *DATA* and *DATA-LEN* are ignored.

Return value

CTBPARAM returns one of the following values:

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.
CS-FAIL (-2)	The routine failed.

Examples

The following code fragment illustrates the use of CTBPARAM. It is taken from the sample program SYCTSAR5 in Appendix B, "Sample RPC Application."

```
 ************************************************************
* INITIATE THE STORED PROCEDURE "SYR2". THE DATA WILL BE    *
* RETURNED FROM THE TABLE SYBASE.SAMPLETB. THIS CAN EITHER  *
* BE A DB2 OR AN Adaptive SERVER TABLE DEPENDING  ON WHETHER*
* THE RPC IS SENT TO A CICS REGION OR A Adaptive SERVER.    *
 ************************************************************

      MOVE LOW-VALUES TO CMDSTR.
      MOVE 4          TO INTARG.
      STRING 'SYR2' DELIMITED BY SIZE INTO CMDSTR.

      CALL 'CTBCOMMA' USING CSL-CMD-HANDLE
                            CSL-RC
                            CS-RPC-CMD
                            CMDSTR
                            INTARG
                            CS-UNUSED.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBCOMMAND failed'
                 DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.

     ****************************
     * SET UP THE RPC PARAMETERS *
     ****************************

      MOVE '@parm1'        TO NM-PARM.
      MOVE 6               TO NMLEN-PARM.
      MOVE CS-FMT-NULLTERM TO FORMT-PARM.
      MOVE CS-RETURN       TO FMTSTATUS-PARM.
      MOVE CS-INT-TYPE     TO DATATYPE-PARM.
      MOVE LENGTH OF PARM1 TO DATALEN.
      MOVE 0               TO PARM1.
```

```
      CALL 'CTBPARAM' USING CSL-CMD-HANDLE
                            CSL-RC
                            DATAFMT-PARM
                            PARM1
                            DATALEN
                            INDIC.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBPARAM CS-INT-TYPE parm1 failed'
                 DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.

      MOVE '@parm2'           TO NM-PARM.
      MOVE 6                  TO NMLEN-PARM.
      MOVE CS-FMT-NULLTERM    TO FORMT-PARM.
      MOVE CS-INPUTVALUE      TO FMTSTATUS-PARM.
      MOVE CS-VARCHAR-TYPE    TO DATATYPE-PARM.
      MOVE PF-DEPT            TO PARR-RET.
      MOVE PF-DEPT-SIZE       TO DATALEN.
      MOVE 255                TO MAXLENGTH-PARM.

      CALL 'CTBPARAM' USING CSL-CMD-HANDLE
                            CSL-RC
                            DATAFMT-PARM
                            PARM2
                            DATALEN
                            INDIC.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBPARAM CS-VARCHAR-TYPE parm2 failed'
                 DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.

*************************************
* SEND THE COMMAND AND THE PARAMETERS *
*************************************
```

```
        CALL 'CTBSEND' USING CSL-CMD-HANDLE
                             CSL-RC.

        IF CSL-RC NOT EQUAL CS-SUCCEED
          THEN
            MOVE SPACES TO MSGSTR
            STRING 'CTBSEND failed'
                   DELIMITED BY SIZE INTO MSGSTR
            PERFORM PRINT-MSG
            PERFORM ALL-DONE
        END-IF.

    SEND-PARAM-EXIT.
        EXIT.
```

Usage

- An application calls CTBCOMMAND to initiate a language request, RPC or message command.
- An application calls CTBPARAM once for each parameter that is sent with the current RPC. It describes each parameter. That description is forwarded to the procedure or transaction called.
- CTBPARAM defines parameters for the following types of commands:
  - Language requests
  - RPCs
- A language request requires input parameter values when the text of the language request contains host variables.
- Parameters must be described by CTBPARAM in the same order in which they are sent to the server. The first CTBPARAM call describes the first parameter, the second CTBPARAM call describes the second parameter, and so on, until all parameters are described and sent.

Defining arguments for language requests

An application calls CTBPARAM with FMT-STATUS as CS-INPUTVALUE to define a parameter value for a language request containing variables.

- A language request can have up to 255 parameters.

The following fields in the DATAFMT structure take special values when describing a parameter for a language request. These are listed in Table 2-14.

Table 2-14: DATAFMT fields for language request parameters with CTBPARAM

Field	Value
NAME	The variable name as it appears in the language string. Transact-SQL names begin with the colon (:) character.
FMT-STATUS	CS-INPUTVALUE
All other fields	Standard CTBPARAM values.

Defining arguments for RPCs

An application calls CTBPARAM with FMT-STATUS as CS-RETURN to define a return parameter for an RPC, and calls CTBPARAM with FMT-STATUS as CS-INPUTVALUE to define a non-return parameter.

- An application can call a stored procedure or transaction in two ways:
  - By sending a language request
  - By issuing an RPC. See “Remote procedure calls (RPCs)” on page 190 for a discussion of the differences between these techniques.
- To send an RPC, a Client-Library application performs the following steps:
  a Calls CTBCOMMAND to initiate the request.
  b Calls CTBPARAM once for each parameter that is being passed to the remote procedure.
  c Calls CTBSEND to send the request to the server. One CTBSEND forwards the RPC with all defined parameters; the application does not call CTBSEND separately for each parameter.
- An RPC can have up to 255 parameters.
- The following fields in the DATAFMT structure take special values when describing an RPC parameter. These are listed in Table 2-15.

Table 2-15: DATAFMT fields for RPC parameters with CTBPARAM

Field	Value
FMT-NAME	When sending parameters to an Adaptive Server, FMT-NAME must begin with the “@” symbol, which prefixes all Adaptive Server stored procedure parameter names.
FMT-MAXLEN	The maximum length of data to be returned by the server. Set to CS-UNUSED if the parameter is non-return, if FMT-TYPE is fixed-length, or if the application does not need to restrict the length of return parameters.
FMT-STATUS	CS-RETURN to indicate that the parameter is a return parameter. CS-INPUTVALUE to indicate that the parameter is not a return parameter.

Field	Value
All other fields	Standard CTBPARAM values.

Table 2-16 lists a summary of arguments for CTBPARAM.

Table 2-16: Summary of arguments (CTBPARAM)

Command	FMT-STATUS value	DATA, DATA-LEN value
Language request	CS-INPUTVALUE	The parameter value and length.
RPC (return parameters)	CS-RETURN	The parameter value and length.
RPC (non-return parameters)	CS-INPUTVALUE	The parameter value and length.

See also

Related functions

- CTBCOMMAND on page 51
- CTBSEND on page 142

CTBREMOTEPWD

Description

Defines or clears passwords to be used for server-to-server connections.

Syntax

```
COPY CTPUBLIC.

01 CONNECTION          PIC S9(9) COMP SYNC.
01 RETCODE             PIC S9(9) COMP SYNC.
01 ACTION              PIC S9(9) COMP SYNC.
01 SERVERNAME          PIC X(30).
01 SRV-LEN             PIC S9(9) COMP SYNC.
01 SRV-BLANKSTRIP      PIC S9(9) COMP SYNC.
01 PASSWD              PIC X(30).
01 PWD-LEN             PIC S9(9) COMP SYNC.
01 PWD-BLANKSTRIP      PIC S9(9) COMP SYNC.

CALL 'CTBREMOT' USING CONNECTION RETCODE ACTION
SERVERNAME SRV-LEN SRV-BLANKSTRIP PASSWD PWD-LEN PWD-
BLANKSTRIP.
```

Parameters

CONNECTION

(I) Handle for this connection. This connection handle must already be allocated with CTBCONALLOC.

Remote passwords can only be defined for a connection before it is open. Passwords defined after a connection is open are ignored.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under “Return value,” in this section.

ACTION

(I) Action to be taken by this call. *ACTION* is an integer variable that indicates the purpose of this call. *ACTION* can be any of the following symbolic values:

Value	Meaning
CS-SET (34)	Sets the remote password.
CS-CLEAR (35)	Clears all remote passwords specified for this connection by assigning LOW-VALUES to SERVERNAME and PASSWD.

SERVERNAME

(I) Name of the server for which the password is being defined. This is the name by which the server is known in the Server Path Table.

If *ACTION* is CS-CLEAR, *SERVERNAME* will default to LOW-VALUES.

If *SERVERNAME* is LOW-VALUES, the specified password will be considered a “universal” password, to be used with any server that does not have a password explicitly specified for it.

SERVERNAME-LEN

(I) Length, in bytes, of *SERVERNAME*. To use the default “universal” password, assign CS-NULL-STRING to this argument. To indicate that the value is terminated at the last non-blank character, assign CS-TRUE to *SRVBLANKSTRIP*.

SRVBLANKSTRIP

(I) Blank termination indicator. Indicates whether the value in the buffer is terminated at the last non-blank character. Assign this argument one of the following symbolic values:

Value	Meaning
CS-TRUE (1)	Trailing blanks are stripped. The value in the buffer ends at the last non-blank character.
CS-FALSE (0)	Trailing blanks are not stripped. They are included in the value.

PASSWD

(I) Password being installed for remote logins to the server named in *SERVERNAME*.

If *ACTION* is CS-CLEAR, PASSWD is passed as LOW-VALUES, and the password defaults to the one set for this connection in CTBCONPROPS, if any.

PASSWD-LEN

(I) Length, in bytes, of *PASSWD*. To indicate that the value is terminated at the last non-blank character, assign CS-TRUE to *PWDBLANKSTRIP*.

PWDBLANKSTRIP

(I) Blank stripping indicator. Indicates whether the value of the password is terminated at the last non-blank character.

Assign this argument one of the following symbolic values:

Value	Meaning
CS-TRUE (1)	Trailing blanks are stripped. The value in the buffer ends at the last non-blank character.
CS-FALSE (0)	Trailing blanks are not stripped. They are included in the value.

Return value

CTBREMOTEPWD returns one of the following values:

Value	Meaning
CS-SUCCEED (-1)	Results are available for processing.
CS-FAIL (-2)	The routine failed.
TDS-INVALID-PARAMETER (-4)	One or more of the CTBREMOTEPWD arguments contains an illegal value. Likely causes for this code are: • Erroneous value for *ACTION*. *ACTION* cannot be CS-GET for CTBREMOTEPWD. • Erroneous value for a length argument. Length values cannot be negative numbers.
TDS-SOS (-257)	Memory shortage. The operation failed.

Examples

The following code fragment demonstrates the use of CTBREMOTEPWD. This sample is not part of any sample program, so the Working Storage section is included here.

```
01 CTX                          PIC S9(9) COMP SYNC VALUE +0.
    01 CON                          PIC S9(9) COMP SYNC VALUE +0.
    01 CMD                          PIC S9(9) COMP SYNC VALUE +0.
```

```
    01 RETCODE                        PIC S9(9) COMP SYNC VALUE +0.
    01 USER                           PIC X(30).
    01 REM-PWD                        PIC X(30).
    01 REM-PWD-LEN                    PIC S9(9) COMP SYNC VALUE IS 0.
    01 STRLEN                         PIC S9(9) COMP SYNC.
    01 SERVNAME                       PIC X(30).
    01 USER-DATA                      PIC X(30).
     01 I                              PIC S9(9) COMP SYNC.
     01 I2                             PIC S9(9) COMP SYNC VALUE IS 0.
     01 DISP-MSG.
        05 TEST-CASE                   PIC X(10) VALUE IS 'RPC SAMPLE'.
        05 FILLER                      PIC X(4) VALUE IS SPACES.
        05 MSG.
           10 SAMP-LIT                 PIC X(3).
           10 SAMP-RC                  PIC -ZZZ9.
           10 FILLER                   PIC X(3) VALUE IS SPACES.
           10 MSGSTR                   PIC X(40) VALUE IS SPACES.

    PROCEDURE DIVISION.
    P0.
* SET THE REMOTE PASSWORD

      MOVE LOW-VALUES TO SERVNAME.
      STRING 'server2' DELIMITED BY SIZE INTO SERVNAME.
      MOVE 7  TO STRLEN.
      STRING 'passwd2' DELIMITED BY SIZE INTO REM-PWD.
      MOVE 7  TO REM-PWD-LEN.
      CALL 'CTBREMOT' USING CON RETCODE CS-SET SERVNAME STRLEN
                      CS-FALSE  REM-PWD REM-PWD-LEN CS-FALSE.
      IF RETCODE NOT EQUAL CS-SUCCEED
          MOVE SPACES TO MSGSTR
           STRING 'CTBREMOT FAILED' DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG.

      MOVE LOW-VALUES TO REM-PWD.
      CALL 'CTBREMOT' USING CON RETCODE CS-GET SERVNAME STRLEN
                      CS-FALSE  REM-PWD REM-PWD-LEN CS-FALSE.
      IF RETCODE NOT EQUAL CS-SUCCEED
          MOVE SPACES TO MSGSTR
           STRING 'CTBREMOT FAILED' DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG.

      MOVE LOW-VALUES TO SERVNAME.
      STRING 'mystring-sun4' DELIMITED BY SIZE INTO SERVNAME.
      MOVE 10 TO STRLEN.
      CALL 'CTBCONNE' USING CON RETCODE SERVNAME STRLEN CS-FALSE.
```

```
IF RETCODE NOT EQUAL CS-SUCCEED
    MOVE SPACES TO MSGSTR
     STRING 'CTBCONNE FAILED' DELIMITED BY SIZE INTO MSGSTR
    PERFORM PRINT-MSG
    PERFORM ALLDONE.
```

Usage

- A Transact-SQL language command, stored procedure, or transaction running on one server can call a stored procedure or transaction located on another server. To accomplish this server-to-server communication, the first server, to which an application connected through CTBCONNECT, actually logs into the second, remote server, performing a server-to-server remote procedure call.

  CTBREMOTEPWD allows an application to specify the password to be used when the first server logs into the remote server.
- Multiple passwords can be specified, one for each server that the first server might need to log into. Each password must be defined with a separate call to CTBREMOTEPWD.
- If an application does not specify a remote password for a particular server, the password defaults to the password set for this connection through CTBCONPROPS, if any. If a password is not defined, the password is set to LOW-VALUES. If an application user generally has the same password on different servers, this default behavior can be sufficient.
- Remote passwords are stored in an internal buffer, which is only 255 bytes long. Each password entry in the buffer consists of the password itself, the associated server name, and two extra bytes. If the addition of a password to this buffer would cause overflow, CTBREMOTEPWD returns CS-FAIL and generates a Client-Library error message that indicates the problem.
- Define remote passwords before calling CTBCONNECT to create an active connection. It is an error to call CTBREMOTEPWD to define a remote password for a connection that is already open.
- An application can call CTBREMOTEPWD to clear remote passwords for a connection at any time.

See also

Related functions

- CTBCONNECT on page 67
- CTBCONPROPS on page 70

CTBRESINFO

Description

Returns result set information.

Syntax

```
COPY CTPUBLIC.

01 COMMAND       PIC S9(9) COMP SYNC.
01 RETCODE       PIC S9(9) COMP SYNC.
01 RESULT-TYP    PIC S9(9) COMP SYNC.
01 BUFFER        PIC S9(9)
01 BUFFER-LEN    PIC S9(9) COMP SYNC.
01 OUTLEN        PIC S9(9) COMP SYNC.

CALL 'CTBRESIN' USING COMMAND RETCODE RESULT-TYP   BUFFER
BUFFER-LEN OUTLEN.
```

Parameters

COMMAND

(I) Handle for this client/server operation. This handle is defined in the associated CTBCMDALLOC call.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under “Return value,” in this section.

RESULT-TYP

(I) Type of information to return. Assign this argument one of the following values:

Value	Meaning
CS-ROW-COUNT (800)	The number of rows affected by the current command.
CS-CMD-NUMBER (801)	The number of the command that generated the current result set.
CS-NUMDATA (803)	The number of items in the current result set.

BUFFER

(O) Variable (“buffer”) where CTBRESINFO returns the requested information.

This argument is typically:

```
01 BUFFER   PIC S9(9) COMP SYNC.
```

BUFFER-LEN

(I) Length, in bytes, of the buffer.

If the returned value is longer than *BUFFER-LEN*, CTBRESINFO sets *OUTLEN* to the length of the requested information and returns CS-FAIL.

OUTLEN
(O) Length, in bytes, of the retrieved information. *OUTLEN* is an integer variable where CTBRESINFO returns the length of the information being retrieved.

If the retrieved information is larger than *BUFFER-LEN* bytes, an application uses the value of *OUTLEN* to determine how many bytes are needed to hold the information.

Return value

CTBRESINFO returns one of the following values:

Value	Meaning
CS-SUCCEED (-1)	Results are available for processing.
CS-FAIL (-2)	The routine failed. CTBRESINFO returns CS-FAIL if the requested information is larger than *BUFFER-LEN* bytes.

Examples

The following code fragment demonstrates the use of CTBRESINFO. It is taken from the sample program SYCTSAA5 in Appendix A, "Sample Language Requests."

```
*============================================================
*==                                                        ==
*== Subroutine to process result rows                      ==
*==                                                        ==
*============================================================
    RESULT-ROW-PROCESSING.

     CALL 'CTBRESIN' USING CSL-CMD-HANDLE,
                           CSL-RC,
                           CS-NUMDATA,
                           RF-NUMDATA,
                           RF-NUMDATA-SIZE,
                           CF-COL-LEN.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBRESINFO failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

     COMPUTE FF-ROW-NUM = FF-ROW-NUM + 1.
```

```
*********************************
* display number of connections *
*********************************

     MOVE CF-MAXCONNECT   TO OR2-MAXCONNECT.
     MOVE OUTPUT-ROW-STR2 TO RSLTNO(FF-ROW-NUM).
     COMPUTE FF-ROW-NUM = FF-ROW-NUM + 2.

*********************************
* display the number of columns *
*********************************

     MOVE RF-NUMDATA      TO OR4-NUMDATA.
     MOVE OUTPUT-ROW-STR4 TO RSLTNO(FF-ROW-NUM).

     IF RF-NUMDATA NOT EQUAL 2
       THEN
         STRING 'CTBRESINFO returned wrong # of parms' DELIMITED
                                          BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

     COMPUTE FF-ROW-NUM = FF-ROW-NUM + 2.
```

Usage

- CTBRESINFO returns information about the current result set or the current command. The current command is defined as the request that generated the current result set.
- A result set is a collection of a single type of result data. Result sets are generated by requests. For more information on result sets, see CTBRESULTS on page 137 and “Results” on page 193.

Retrieving the command number for the current result set

To determine the number of the command that generated the current result set, call CTBRESINFO with *RESULT-TYP* as CS-CMD-NUMBER.

- Client-Library keeps track of the command number by counting the number of times CTBRESULTS returns CS-CMD-DONE.

  An application’s first call to CTBRESULTS following a CTBSEND call sets the command number to 1. The command number remains 1 until CTBRESULTS returns CS-CMD-DONE. The next time the application calls CTBRESULTS, the command number is incrementally increased to 2. The command number continues to be increased by 1 each time CTBRESULTS is called after returning CS-CMD-DONE.

- CS-CMD-NUMBER is useful in the following cases:
  - To determine the SQL command within a language request that generated the current result set.
  - To determine the select command in a stored procedure or transaction that generated the current result set.
- A language request contains a string of text. This text represents one or more SQL commands or other language request statements. If the application is sending a language request, "command number" refers to the number of the statement in the language request.

  For example, the following Transact-SQL string represents three Transact-SQL commands—two select statements and one insert:

```
select * from authors
 select * from titles
 insert newauthors
 select * from authors
 where city = "San Francisco"
```

  The two select statements can generate result sets. In this case, the command number that CTBRESINFO returns can be from 1 to 3, depending on when CTBRESINFO is called.

  Note When sending SQL strings to DB2, remember to use semicolons (;) to separate SQL statements.

- Inside stored procedures or transactions, only select statements cause the command number to be incremented. If a stored procedure or transaction contains seven SQL commands, three of which are select statements, the command number that CTBRESINFO returns can be any integer from 1 to 3, depending on which select statement generated the current result set.

Retrieving the number of result data items

To determine the number of result data items in the current result set, call CTBRESINFO with *RESULT-TYP* as CS-NUMDATA.

- Results sets contain result data items. Row result sets contain columns, a parameter result set contains parameters, and a status result set contains a status. The columns, parameters, and status are known as result data items.

Retrieving the number of rows for the current command

To determine the number of rows affected by the current command, call CTBRESINFO with *RESULT-TYP* as CS-ROW-COUNT.

- If the current command is one that does not return rows—for example, a language command containing an insert statement—an application can get the row count immediately after CTBRESULTS returns CS-CMD-SUCCEED.
- If the current command does return rows:
  - An application can get a total row count after processing all of the rows.
  - An application can get an intermediate row count any time after CTBRESULTS indicates that results are available. An intermediate row count is equivalent to the number of rows that have been fetched so far.
- If the command is one that executes a stored procedure or transaction—for example a Transact-SQL exec language command or a remote procedure call—CTBRESINFO returns either the number of rows returned by the latest select statement executed by the stored procedure or transaction, or CS-NO-COUNT if the stored procedure or transaction does not execute any select statements. A stored procedure or transaction that does not contain any select statements can execute a select by calling another stored procedure or transaction that contains a select statement.
- CTBRESINFO returns CS-NO-COUNT if any of the following are true:
  - The SQL command fails for any reason, such as a syntax error.
  - The command is one that *never* affects rows, such as a Transact-SQL print command.
  - The command executes a stored procedure or transaction that does not execute any select statements.

The following arguments listed in Table 2-17 on page 136 are returned to *RESULT-TYP* after CTBRESULTS indicates that results are present.

Table 2-17: Summary of arguments (CTBRESINFO)

RESULT-TYP	CTBRESINFO returns	BUFFER value
CS-ROW-COUNT (800)	The number of rows affected by the current command.	An integer value.
CS-CMD-NUMBER (801)	The number of the command that generated the current result set.	An integer value.
CS-NUMDATA (803)	The number of items in the current result set.	An integer value.

See also

Related functions

- CTBCMDPROPS on page 47
- CTBCONPROPS on page 70
- CTBRESULTS on page 137

CTBRESULTS

Description

Sets up result data to be processed.

Syntax

```
COPY CTPUBLIC.

01 COMMAND        PIC S9(9) COMP SYNC.
01 RETCODE        PIC S9(9) COMP SYNC.
01 RESULT-TYP     PIC S9(9) COMP SYNC.

CALL 'CTBRESUL' USING COMMAND RETCODE RESULT-TYP.
```

Parameters

COMMAND

(I) Handle for this client/server operation. This handle is defined in the associated CTBCMDALLOC call.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under “Return value,” in this section.

RESULT-TYP

(O) Variable containing the result type. CTBRESULTS returns to this variable a symbolic value that indicates the type of result returned by the current request. The result type can be any of the following symbolic values listed in Table 2-18.

Table 2-18: Values for RESULT-TYP (CTBRESULTS)

Value	Meaning	Result produced
CS-ROW-RESULT (4040)	Regular row results arrived.	One or more rows of tabular data.
CS-PARAM-RESULT (4042)	Return parameter results arrived.	A single row of return parameters.
CS-STATUS-RESULT (4043)	Stored procedure return status results arrived.	A single row containing a single status.
CS-CMD-DONE (4046)	The results of the request processed completely.	Not applicable.

Value	Meaning	Result produced
CS-CMD-SUCCEED (4047)	A request that returns no data, such as a language request containing an insert statement, processed successfully.	No results.
CS-CMD-FAIL (4048)	The server encountered an error while executing the request. This value can indicate that the connection failed or was terminated.	No results.

Return value

CTBRESULTS returns one of the following values:

Value	Meaning
CS-SUCCEED (-1)	A result set is available for processing.
CS-END-RESULTS (-205)	No more result sets are available for processing.
CS-FAIL (-2)	The routine failed.
CS-CANCELLED (-202)	Results were cancelled.

Examples

The following code fragment demonstrates how CTBRESULTS can describe a result row for a language request. It is taken from the sample program SYCTSAA5 in Appendix A, “Sample Language Requests.”

```
 *=========================================================
 *==                                                     ==
 *== Subroutine to process result                        ==
 *==                                                     ==
 *=========================================================
RESULTS-PROCESSING.
    ***************************
    * SET UP THE RESULTS DATA *
    ***************************
         CALL 'CTBRESUL' USING CSL-CMD-HANDLE
                               CSL-RC
                               RESTYPE.
    ************************************************
    * DETERMINE THE OUTCOME OF THE COMMAND EXECUTION *
    ************************************************
         EVALUATE CSL-RC
         WHEN CS-SUCCEED
    ************************************************************
    * DETERMINE THE TYPE OF RESULT RETURNED BY THE CURRENT REQUEST *
    ************************************************************
                 EVALUATE RESTYPE
    ***********************
    * PROCESS ROW RESULTS *
    ***********************
```

```
                WHEN CS-ROW-RESULT
                MOVE LOW-VALUES TO A5PANELO
                PERFORM RESULT-ROW-PROCESSING
                MOVE 'Y' TO SW-FETCH
                PERFORM FETCH-ROW-PROCESSING UNTIL NO-MORE-ROWS
*************************************************************
* PROCESS PARAMETER RESULTS - THERE SHOULD BE NO PARAMETERS *
* TO PROCESS                                                *
*************************************************************
                WHEN CS-PARAM-RESULT
                  MOVE 'Y' TO SW-FETCH
*************************************************************
* PROCESS STATUS RESULTS - THE STORED PROCEDURE STATUS RESULT *
* WILL NOT BE PROCESSED IN THIS EXAMPLE                      *
*************************************************************
                WHEN CS-STATUS-RESULT
                  MOVE 'Y' TO SW-FETCH
*************************************************************
* PRINT AN ERROR MESSAGE IF THE SERVER ENCOUNTERED AN ERROR *
* WHILE EXECUTING THE REQUEST                               *
*************************************************************
                WHEN CS-CMD-FAIL
                  STRING
                     'CTBRESUL returned CS-CMD-FAIL restype'
                     DELIMITED BY SIZE INTO MSGSTR
                  PERFORM PRINT-MSG
*************************************************************
* PRINT A MESSAGE FOR SUCCESSFUL COMMANDS THAT RETURNED NO DATA *
* (OPTIONAL)                                                    *
*************************************************************
                WHEN CS-CMD-SUCCEED
                  STRING
                     'CTBRESUL returned CS-CMD-SUCCEED restype'
                     DELIMITED BY SIZE INTO MSGSTR
*********************************************************
* PRINT A MESSAGE FOR REQUESTS THAT HAVE BEEN PROCESSED *
* SUCCESSFULLY (OPTIONAL)                               *
*********************************************************
                WHEN CS-CMD-DONE
                  STRING 'CTBRESUL returned CS-CMD-DONE restype'
                            DELIMITED BY SIZE INTO MSGSTR
                WHEN OTHER
                   STRING 'CTBRESUL returned UNKNOWN restype'
                            DELIMITED BY SIZE INTO MSGSTR
                   PERFORM PRINT-MSG
                   MOVE 'N' TO SW-RESULTS
```

```
                END-EVALUATE
           ******************************************************
           * PRINT AN ERROR MESSAGE IF THE CTBRESULTS CALL FAILED *
           ******************************************************
                    WHEN CS-FAIL
                      MOVE 'N' TO SW-RESULTS
                      STRING 'CTBRESUL returned CS-FAIL ret-code'
                                DELIMITED BY SIZE INTO MSGSTR
                      PERFORM PRINT-MSG
      ************************************************************
           * DROP OUT OF THE RESULTS LOOP IF NO MORE RESULT SETS ARE   *
           * AVAILABLE FOR PROCESSING OR IF THE RESULTS WERE CANCELLED *
           ************************************************************
                    WHEN CS-END-RESULTS
                      MOVE 'N' TO SW-RESULTS
                    WHEN CS-CANCELLED
                      MOVE 'N' TO SW-RESULTS
                     WHEN OTHER
                      MOVE 'N' TO SW-RESULTS
                      STRING 'CTBRESUL returned UNKNOWN ret-code'
                                DELIMITED BY SIZE INTO MSGSTR
                      PERFORM PRINT-MSG
                END-EVALUATE.
                MOVE 0 TO RESTYPE.
            RESULTS-PROCESSING-EXIT.
                EXIT.
```

Usage

- CTBRESULTS tells the application what kind of results returned and sets up result data for processing. An application calls CTBRESULTS after sending a request to the server through CTBSEND and before binding and retrieving the results of that request (if any) with CTBBIND and CTBFETCH.
- "Result data" is an umbrella term for all the types of data that a server can return to an application:
  - Regular rows
  - Return parameters
  - Stored procedure return status

  CTBRESULTS is used to set up all of these types of results for processing.
- Result data is returned to an application in the form of result sets. A result set includes only a single type of result data. For example, a regular row result set contains only regular rows, and a return parameter result set contains only return parameters.

The *CTBRESULTS* loop

- Because a request can generate multiple result sets, an application must call CTBRESULTS as long as it continues to return CS-SUCCEED, indicating that results are available. The simplest way to do this is in a loop that terminates when CTBRESULTS fails to return CS-SUCCEED. After the loop, an application can test the CTBRESULTS final return code to determine why the loop terminated.
- Results are returned to an application in the order in which they are produced. However, this order is not always easy to predict. For example, when an application calls a stored procedure or transaction that in turn calls another stored procedure or transaction, the application might receive a number of row result sets, as well as a return parameter and a return status result set. The order in which these results are returned depends on how the called stored procedure or transaction is written.

  For this reason, we recommend that you include a series of IF statements in your application, ending with a statement that handles all types of results that can be received.

  The *RESULT-TYP* argument indicates what type of result data the result set contains.

When are the results of a command completely processed?

- CTBRESULTS sets the result type to CS-CMD-DONE to indicate that the results of a logical command processed completely.

  A logical command is any command defined through CTBCOMMAND, with the following rules:

  - Each Transact-SQL select statement inside a stored procedure is a logical command. Other Transact-SQL statements inside stored procedures do not count as logical commands.
  - Each Transact-SQL statement in a language request is a logical command.
- A result type of CS-CMD-SUCCEED or CS-CMD-FAIL is immediately followed by a result type of CS-CMD-DONE.

Canceling results

To cancel remaining results from a request (and eliminate the need to continue calling CTBRESULTS until it fails to return CS-SUCCEED), call CTBCANCEL.

CTBRESULTS and stored procedures

A run-time error on a language request containing an execute statement returns CS-CMD-FAIL. However, a run-time error on a statement inside a stored procedure or transaction does not return CS-CMD-FAIL. For example, if a called stored procedure or transaction contains an insert statement and the user does not have insert permission on the database table, the insert statement fails, but CTBRESULTS still returns CS-SUCCEED.

If results are coming from Open ServerConnect, a return status of TDS-DONE-ERROR indicates an error.

See also

Related functions

- CTBBIND on page 25
- CTBCOMMAND on page 51
- CTBDESCRIBE on page 78
- CTBFETCH on page 107
- CTBSEND on page 142

Related topics

- “Remote procedure calls (RPCs)” on page 190
- “Results” on page 193

CTBSEND

Description

Sends a request to the server.

Syntax

```
COPY CTPUBLIC.

01 COMMAND        PIC S9(9) COMP SYNC.
01 RETCODE        PIC S9(9) COMP SYNC.

CALL 'CTBSEND' USING COMMAND RETCODE.
```

Parameters

COMMAND

(I) Handle for this client/server operation. This handle is defined in the associated CTBCMDALLOC call.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under “Return value,” in this section.

Return value CTBSEND returns one of the following values:

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.
CS-FAIL (-2)	The routine failed. This result can indicate that SNA sessions will not come up.
CS-CANCELLED (-202)	The routine was cancelled. **Note** This value is returned by SNA sessions only, and is never returned when sending a request to another CICS region.

Examples **Example 1**

The following code fragment demonstrates the use of CTBSEND. It is taken from the sample program SYCTSAR5 in Appendix B, “Sample RPC Application.”

```
*=========================================================
*==                                                     ==
*== Subroutine to allocate, send, and process commands ==
*==                                                     ==
*=========================================================
 SEND-PARAM.
*****************************
* NOW GET A COMMAND HANDLE. *
*****************************
     MOVE ZERO TO CSL-CMD-HANDLE.
     CALL 'CTBCMDAL' USING CSL-CON-HANDLE
                           CSL-RC
                           CSL-CMD-HANDLE.
     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBCMDAL failed'
                DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

*************************************************************
* INITIATE THE STORED PROCEDURE "SYR2". THE DATA WILL BE    *
* RETURNED FROM THE TABLE SYBASE.SAMPLETB. THIS CAN EITHER *
* BE A DB2 OR A Adaptive SERVER TABLE DEPENDING  ON WHETHER*
* THE RPC IS SENT TO A CICS REGION OR A Adaptive SERVER.    *
*************************************************************
     MOVE LOW-VALUES TO CMDSTR.
     MOVE 4          TO INTARG.
```

```
      STRING 'SYR2' DELIMITED BY SIZE INTO CMDSTR.
      CALL 'CTBCOMMA' USING CSL-CMD-HANDLE
                            CSL-RC
                            CS-RPC-CMD
                            CMDSTR
                            INTARG
                            CS-UNUSED.
      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBCOMMAND failed'
                 DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.
*****************************
* SET UP THE RPC PARAMETERS *
*****************************
      MOVE '@parm1'         TO NM-PARM.
      MOVE 6                TO NMLEN-PARM.
      MOVE CS-FMT-NULLTERM  TO FORMT-PARM.
      MOVE CS-RETURN        TO FMTSTATUS-PARM.
      MOVE CS-INT-TYPE      TO DATATYPE-PARM.
      MOVE LENGTH OF PARM1  TO DATALEN.
      MOVE 0                TO PARM1.
      CALL 'CTBPARAM' USING CSL-CMD-HANDLE
                            CSL-RC
                            DATAFMT-PARM
                            PARM1
                            DATALEN
                            INDIC.
      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBPARAM CS-INT-TYPE parm1 failed'
                 DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.
      MOVE '@parm2'            TO NM-PARM.
      MOVE 6                   TO NMLEN-PARM.
      MOVE CS-FMT-NULLTERM     TO FORMT-PARM.
      MOVE CS-INPUTVALUE       TO FMTSTATUS-PARM.
      MOVE CS-VARCHAR-TYPE     TO DATATYPE-PARM.
      MOVE PF-DEPT             TO PARR-RET.
      MOVE PF-DEPT-SIZE        TO DATALEN.
```

```
     MOVE 255                TO MAXLENGTH-PARM.
     CALL 'CTBPARAM' USING CSL-CMD-HANDLE
                           CSL-RC
                           DATAFMT-PARM
                           PARM2
                           DATALEN
                           INDIC.
     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBPARAM CS-VARCHAR-TYPE parm2 failed'
                DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.
************************************
* SEND THE COMMAND AND THE PARAMETERS *
************************************
     CALL 'CTBSEND' USING CSL-CMD-HANDLE
                          CSL-RC.
     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBSEND failed'
                DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.
 SEND-PARAM-EXIT.
     EXIT.
```

Example 2

The following code fragment demonstrates the use of ct_send. It is taken from the sample program SYCTSAA5 in Appendix A, “Sample Language Requests.”

```
      *============================================================
      *==                                                        ==
      *== Subroutine to allocate, send, and process commands ==
      *==                                                        ==
      *============================================================
       SEND-COMMAND.
      *------------------------------------------------------------
      *  find out what the maximum number of connections is
      *------------------------------------------------------------
           CALL 'CTBCONFI'  USING CSL-CTX-HANDLE,
```

```
                              CSL-RC,
                              CS-GET,
                              CS-MAX-CONNECT,
                              CF-MAXCONNECT,
                              CF-FOUR,
                              CS-FALSE,
                              CF-OUTLEN.
     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBCONFI CS-GET failed' DELIMITED BY SIZE
                                                  INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.
 *-------------------------------------------------------------
 *  allocate a command handle
 *-------------------------------------------------------------
     CALL 'CTBCMDAL' USING CSL-CON-HANDLE,
                           CSL-RC,
                           CSL-CMD-HANDLE.
     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBCMDAL failed' DELIMITED BY SIZE
                                                  INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

*-------------------------------------------------------------
 *    prepare the language request
 *-------------------------------------------------------------
     MOVE CF-LANG2-SIZE TO PF-STRLEN.
     CALL 'CTBCOMMA' USING CSL-CMD-HANDLE,
                           CSL-RC,
                           CS-LANG-CMD,
                           CF-LANG2,
                           PF-STRLEN,
                           CS-UNUSED.
     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBCOMMA CS-LANG-CMD failed' DELIMITED BY SIZE
                                                  INTO MSGSTR
         PERFORM PRINT-MSG
```

```
                PERFORM ALL-DONE
              END-IF.

    *-------------------------------------------------------------
    *    send the language request
    *-------------------------------------------------------------
              CALL 'CTBSEND' USING CSL-CMD-HANDLE,
                                   CSL-RC.
              IF CSL-RC NOT EQUAL CS-SUCCEED
                THEN
                  MOVE SPACES TO MSGSTR
                  STRING 'CTBSEND failed' DELIMITED BY SIZE
                                                      INTO MSGSTR
                  PERFORM PRINT-MSG
                  PERFORM ALL-DONE
              END-IF.
            SEND-COMMAND-EXIT.
              EXIT.
```

Usage

- CTBSEND signals the end of the data to be sent to a server (no more parameters, data, messages) and sends a request to the server.
- Sending a request to a server is a three-step process. To send a request to a server, an application:
  - Initiates the request by calling CTBCOMMAND, which initiates a language request, RPC, or message stream to send to the server.
  - Describes parameters for the request, using CTBPARAM.

    Not all requests require parameters. For example, a remote procedure call may or may not require parameters, depending on the stored procedure or transaction being called.
  - Calls CTBSEND to send the request stream to the server.
- CTBSEND does not wait for a response from the server. An application must call CTBRESULTS to verify the success of the request and to set up the results for processing.

See also

Related functions

- CTBCOMMAND on page 51
- CTBFETCH on page 107
- CTBPARAM on page 120

- CTBRESULTS on page 137

CSBCONFIG

Description

Sets or retrieves context structure properties.

Syntax

```
COPY CTPUBLIC.

01 CONTEXT PIC S9(9) COMP SYNC.
01 RETCODE PIC S9(9) COMP SYNC.
01 ACTION PIC S9(9) COMP SYNC.
01 PROPERTY PIC S9(9) COMP SYNC.
01 BUFFER PIC X(n)
01 BUFFER-LEN PIC S9(9) COMP SYNC.
01 BUFBLANKSTRIP PIC S9(9) COMP SYNC.
01 OUTLEN PIC S9(9) COMP SYNC.

CALL 'CSBCONFI' USING CONTEXT RETCODE OPTION ACTION
BUFFER BUFFER-LEN BUFBLANKSTRIP OUTLEN.
```

Parameters

CONTEXT

(I) A context structure for which the properties are being set or retrieved. The context structure is defined in the program call CSBCTXALLOC.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under “Return value,” in this section.

ACTION

(I) Action to be taken by this call. *ACTION* is an integer variable that indicates the purpose of this call. Assign *ACTION* one of the following symbolic values:

Value	Meaning
CS-GET (33)	Retrieves the value of the property.
CS-SET (34)	Sets the value of the property.
CS-CLEAR (35)	Clears the value of the property by resetting the property to its default value.

PROPERTY

(I) Symbolic name of the property for which the value is being set or retrieved. Client-Library properties are listed under "Properties" on page 181, with descriptions, possible values, and defaults. Table 2-19 on page 149 lists the properties that can be set or retrieved by CSBCONFIG.

Table 2-19: Values for PROPERTY (CSBCONFIG)

Application action	Property	Indicates
Set, retrieve, or clear	CS-EXTRA-INF	Whether to return the extra information required when processing messages in line, using the SQLCA or SQLCODE structures.
Retrieve only	CS-VERSION	The version number of Open Client currently in use.

BUFFER

(I/O) Variable (buffer) that contains the specified property value.

If *ACTION* is CS-SET, CSBCONFIG takes the value from this buffer.

If *ACTION* is CS-GET, CSBCONFIG returns the requested information to this buffer.

If *ACTION* is CS-CLEAR, the buffer is reset to the default property value.

This argument is typically one of the following datatypes:

```
01 BUFFER PIC S9(9) COMP SYNC.
01 BUFFER PIC X(n).
```

BUFFER-LEN

(I) Length, in bytes, of the buffer.

If *ACTION* is CS-SET and the value in the buffer is a fixed-length or symbolic value, *BUFFER-LEN* should have a value of CS-UNUSED. To indicate that the terminating character is the last non-blank character, an application sets *BUFBLANKSTRIP* to CS-TRUE.

If *ACTION* is CS-GET and *BUFFER* is too small to hold the requested information, CSBCONFIG sets *OUTLEN* to the length of the requested information and returns CS-FAIL. To retrieve all the requested information, change the value of *BUFFER-LEN* to the length returned in *OUTLEN* and rerun the application.

If *ACTION* is CS-CLEAR, this value is ignored.

BUFBLANKSTRIP

(I) Blank stripping indicator. Indicates whether trailing blanks are stripped.

Assign this argument one of the following symbolic values:

Value	Meaning
CS-TRUE (1)	Trailing blanks are stripped. The value in the buffer ends at the last non-blank character.
CS-FALSE (0)	Trailing blanks are not stripped. They are included in the value.

If a property value is being set and the terminating character is the last non-blank character, assign CS-TRUE to *BUFBLANKSTRIP*.

OUTLEN

(O) Length, in bytes, of the retrieved information. *OUTLEN* is an integer variable where CSBCONFIG returns the length of the property value being retrieved.

When the retrieved information is larger than *BUFFER-LEN* bytes, an application uses the value of *OUTLEN* to determine how many bytes are needed to hold the information.

OUTLEN is used only when *ACTION* is CS-GET. When the *ACTION* is CS-SET or CS-CLEAR, this value is ignored.

Return value

CSBCONFIG returns one of the following values:

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.
CS-FAIL (-2)	The routine failed.
TDS-INVALID-PARAMETER (-4)	One of the CSBCONFIG arguments contains an illegal value. The most likely cause for this code is that a property value is being set and the value assigned to BUFBLANKSTRIP is not CS-TRUE.

Examples

This code fragment demonstrates the use of CSBCONFIG. It is not taken from any of the sample programs.

```
01 CTX                          PIC S9(9) COMP SYNC VALUE +0.
01 CON                          PIC S9(9) COMP SYNC VALUE +0.
01 CMD                          PIC S9(9) COMP SYNC VALUE +0.
01 RET                          PIC S9(9) COMP SYNC VALUE +0.
01 RETCODE                      PIC S9(9) COMP SYNC VALUE +0.
01 VERSION                      PIC S9(9) COMP SYNC VALUE IS 0.
01 INF-VAL                      PIC S9(9) COMP SYNC VALUE IS 0.
```

```
01 DISP-ROW.
   05 ROW1-VAL                    PIC X(15) VALUE IS SPACES.
   05 ROW2-VAL                    PIC X(8)  VALUE IS SPACES.
   05 FILLER                      PIC X(1)  VALUE IS SPACES.
   05 ROW3-VAL                    PIC X(9)  VALUE IS SPACES.
   05 FILLER                      PIC X(4) VALUE IS SPACES.
   05 ROW4-VAL.
       49   HIGH-VAL              PIC  ZZZ,ZZZ,ZZZ.
       49   LOW-VAL               PIC  ZZ,ZZZ.99-.
   05 FILLER                      PIC X(21) VALUE IS SPACES.

01 OUTLENPIC S9(9) COMP SYNC.
01 DISP-MSG.
      05 TEST-CASE                PIC X(10) VALUE IS 'RPC SAMPLE'.
      05 FILLER                   PIC X(4) VALUE IS SPACES.
      05 MSG.
            10 SAMP-LIT           PIC X(3).
            10 SAMP-RC            PIC -ZZZ9.
            10 FILLER             PIC X(3) VALUE IS SPACES.
            10 MSGSTR             PIC X(40) VALUE IS SPACES.
01 DATAFMT-PARM.
      05 NM-PARM                  PIC X(132).
      05 NMLEM-PARM               PIC S9(9) COMP SYNC.
      05 DATATYPE-PARM            PIC S9(9) COMP SYNC.
      05 FORMT-PARM               PIC S9(9) COMP SYNC.
      05 MAXLENGTH-PARM           PIC S9(9) COMP SYNC.
      05 SCALE-PARM               PIC S9(9) COMP SYNC.
      05 PRECISION-PARM           PIC S9(9) COMP SYNC.
      05 FNTSTATUS-PARM           PIC S9(9) COMP SYNC.
      05 FMTCOUNT-PARM            PIC S9(9) COMP SYNC.
      05 USERTYPE-PARM            PIC S9(9) COMP SYNC.
      05 LOCALE-PARM              PIC S9(9) COMP SYNC.
01 DATAFMT-BIND.
      05 NM-BIND                  PIC X(132).
      05 NMLEN-BIND               PIC S9(9) COMP SYNC.
      05 DATATYPE-BIND            PIC S9(9) COMP SYNC.
      05 FORMT-BIND               PIC S9(9) COMP SYNC.
      05 MAXLENGTH-BIND           PIC S9(9) COMP SYNC.
      05 SCALE-BIND               PIC S9(9) COMP SYNC.
      05 PRECISION-BIND           PIC S9(9) COMP SYNC.
      05 FMTSTATUS-BIND           PIC S9(9) COMP SYNC.
      05 FMTCOUNT-BIND            PIC S9(9) COMP SYNC.
      05 USERTYPE-BIND            PIC S9(9) COMP SYNC.
      05 LOCALE-BIND              PIC S9(9) COMP SYNC.

PROCEDURE DIVISION.
```

```
      P0.
      * ALLOCATE A CONTEXT STRUCTURE
        MOVE ZERO TO CTX.
        MOVE LOW-VALUES TO DATAFMT-PARM DATAFMT-BIND DISP-ROW.
        CALL 'CSBCTXAL' USING CS-VERSION-100 RETCODE CTX.
        IF RETCODE NOT EQUAL CS-SUCCEED
             MOVE SPACES TO MSGSTR
             STRING 'CSBCTXAL FAILED' DELIMITED BY SIZE INTO MSGSTR
             PERFORM PRINT-MSG
             PERFORM ALLDONE.

    * SET THE CONTEXT STRUCTURE PROPERTY CS-EXTRA-INF

        CALL 'CSBCONFI'   USING CTX RETCODE CS-SET CS-EXTRA-INF
                          CS-TRUE  CS-UNUSED CS-FALSE OUTLEN.
        IF RETCODE NOT EQUAL CS-SUCCEED
             MOVE SPACES TO MSGSTR
             STRING 'CSBCONFIG FAILED' DELIMITED BY SIZE INTO MSGSTR
             PERFORM PRINT-MSG.

        CALL 'CSBCONFI'   USING CTX RETCODE CS-GET CS-EXTRA-INF
                          INF-VAL  CS-UNUSED CS-FALSE OUTLEN.
        IF RETCODE NOT EQUAL CS-SUCCEED
             MOVE SPACES TO MSGSTR
             STRING 'CSBCONFIG FAILED' DELIMITED BY SIZE INTO MSGSTR
             PERFORM PRINT-MSG.

        IF INF-VAL NOT EQUAL CS-TRUE
             MOVE SPACES TO MSGSTR
             STRING 'CSBCONFIG RETURNED THE WRONG VALUE'
                   DELIMITED BY SIZE INTO MSGSTR.

        CALL 'CSBCONFI'   USING CTX RETCODE CS-GET CS-VERSION
                          VERSION CS-UNUSED CS-FALSE OUTLEN.
        IF RETCODE NOT EQUAL CS-SUCCEED
             MOVE SPACES TO MSGSTR
             STRING 'CSBCONFIG FAILED' DELIMITED BY SIZE INTO MSGSTR
             PERFORM PRINT-MSG.

        IF VERSION NOT EQUAL CS-VERSION-100
             MOVE SPACES TO MSGSTR
             STRING 'CSBCONFIG RETURNED THE WRONG VERSION'
                   DELIMITED BY SIZE INTO MSGSTR.

             PERFORM PRINT-MSG.
```

Usage

Note CSBCONFIG and CTBCONFIG both set and retrieve context properties. CSBCONFIG is used with global context properties; CTBCONFIG is used with Client-Library properties.

- CSBCONFIG can be used to set and retrieve the value of CS-EXTRA-INF and to retrieve the version number of Open Client currently in use.
- Use CTBCONFIG to set and retrieve the values of Client-Library-specific context properties. Properties set through CTBCONFIG affect only Client-Library behaviors.

Extra information

- CS-EXTRA-INF determines whether or not Client-Library returns the extra information that is required to fill in a SQLCA or SQLCODE structure.
- If an application is not retrieving messages into a SQLCA or SQLCODE structure, the extra information is returned as ordinary Client-Library messages.

Version level

- The CS-VERSION property represents the version of Client-Library behavior that an application requests through CSBCTXALLOC.
- An application can only retrieve the value of CS-VERSION; it cannot assign a value to CS-VERSION.

See also

Related functions

- CSBCTXALLOC on page 160
- CTBCONPROPS on page 70
- CTBCONFIG on page 64
- CTBINIT on page 117

CSBCONVERT

Description

Converts a data value from one datatype to another.

Syntax

COPY CTPUBLIC.

```
01 CONTEXT       PIC S9(9) COMP SYNC.
01 RETCODE       PIC S9(9) COMP SYNC.
01 SRCFMT
   05 SRC-NAME        PIC X(132).
   05 SRC-NAMELEN     PIC S9(9) COMP SYNC.
   05 SRC-TYPE        PIC S9(9) COMP SYNC.
   05 SRC-FORMAT      PIC S9(9) COMP SYNC.
   05 SRC-MAXLEN      PIC S9(9) COMP SYNC.
   05 SRC-SCALE       PIC S9(9) COMP SYNC.
   05 SRC-PRECIS      PIC S9(9) COMP SYNC.
   05 SRC-STATUS      PIC S9(9) COMP SYNC.
   05 SRC-COUNT       PIC S9(9) COMP SYNC.
   05 SRC-UTYPE       PIC S9(9) COMP SYNC.
   05 SRC-LOCALE      PIC S9(9) COMP SYNC.
01 SRCDATA        type.
01 DESTFMT
   05 DEST-NAME       PIC X(132).
   05 DEST-NAMELEN    PIC S9(9) COMP SYNC.
   05 DEST-TYPE       PIC S9(9) COMP SYNC.
   05 DEST-FORMAT     PIC S9(9) COMP SYNC.
   05 DEST-MAXLEN     PIC S9(9) COMP SYNC.
   05 DEST-SCALE      PIC S9(9) COMP SYNC.
   05 DEST-PRECIS     PIC S9(9) COMP SYNC.
   05 DEST-STATUS     PIC S9(9) COMP SYNC.
   05 DEST-COUNT      PIC S9(9) COMP SYNC.
   05 DEST-UTYPE      PIC S9(9) COMP SYNC.
   05 DEST-LOCALE     PIC S9(9) COMP SYNC.
01 DESTDATA       type.
01 OUTLEN           PIC S9(9) COMP SYNC.

CALL 'CSBCONVE' USING CONTEXT RETCODE SRCFMT SRCDATA
DESTFMT DESTDATA OUTLEN.
```

Parameters

CONTEXT

(I) A context structure. The context structure is defined in the program call CSBCTXALLOC.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under “Return value,” in this section.

SRCFMT

(I) A structure that describes the variable(s) that contain the source data. CSBCONVERT ignores *SRCFMT* fields that it does not use.

Table 2-20 lists the fields in the *SRCFMT* structure and indicates whether and how they are used by CSBCONVERT. For a general discussion of this structure, see “DATAFMT structure” on page 170.

Table 2-20: Fields in the SRCFMT structure for CSBCONVERT

Field	When used	Value represents
SRC-NAME	Not used (CS-FMT-UNUSED).	Not applicable.
SRC-NAMELEN	Not used (CS-FMT-UNUSED).	Not applicable.
SRC-TYPE	For all datatype conversions.	The datatype of the source data. CSBCONVERT converts this datatype to the datatype specified for the destination variable (*DEST-TYPE*).
SRC-FORMAT	Not used (CS-FMT-UNUSED).	Not applicable.
SRC-MAXLEN	When converting non-fixed-length source datatypes to any destination type. SRC-MAXLEN is ignored when converting fixed-length types.	The length of the source variable, in bytes. If SRCDATA is an array, SRC-MAXLEN is the length of an element in the array. When converting character or binary datatypes, SRC-MAXLEN must describe the total length of the source variable, including any space required for special terminating bytes, with this exception: when converting a VARYCHAR-type source such as the DB2 VARCHAR, SRC-MAXLEN does not include the length of the "LL" length specification. In case of Sybase-numeric, Sybase-decimal or packed decimal this value is the actual length.
SRC-SCALE	Only when converting to or from numeric, Sybase-decimal, or packed decimal datatypes.	Number of digits that follow the decimal point in the source data. SRC-SCALE must be less than or equal to SRC-PRECIS and cannot be greater than 31.
SRC-PRECIS	Only when converting to or from packed decimal, numeric and Sybase-decimal datatypes.	The total number of digits in the source data. SRC-PRECIS must be greater than or equal to SRC-SCALE and cannot be less than 1 or greater than 31.
SRC-STATUS	Not used (CS-FMT-UNUSED).	Not applicable.
SRC-COUNT	Not used (CS-FMT-UNUSED).	Not applicable.
SRC-UTYPE	Not used (CS-FMT-UNUSED).	Not applicable.
SRC-LOCALE	Not used (CS-FMT-UNUSED).	Not applicable.

SRCDATA

(I) Name of the source variable that contains the data to be converted. This is the variable described in the SRCFMT structure.

DESTFMT

(I) A structure that contains a description of the variable(s) that contain destination (converted) data. CSBCONVERT ignores DESTFMT fields that it does not use.

Table 2-21 lists the fields in the DESTFMT structure and indicates whether and how they are used by CSBCONVERT. For a general discussion of this structure, see "DATAFMT structure" on page 170.

Table 2-21: Fields in the DATAFMT structure for CSBCONVERT

Field	When used	Value represents
DEST-NAME	Not used (CS-FMT-UNUSED).	Not applicable.
DEST-NAMELEN	Not used (CS-FMT-UNUSED).	Not applicable.
DEST-TYPE	For all datatype conversions.	The datatype of the destination variable. CSBCONVERT converts the datatype specified for the source data (SRCTYPE) to this datatype.
DEST-FORMAT	Not used (CS-FMT-UNUSED).	Not applicable.
DEST-MAXLEN	When converting all source datatypes to non-fixed-length datatypes. DEST-MAXLEN is ignored when converting to fixed-length datatypes.	The length of the destination variable, in bytes. If DESTDATA is an array, DEST-MAXLEN is the length of an element in the array. When converting character or binary datatypes, DEST-MAXLEN must describe the total length of the destination variable, including any space required for special terminating bytes, with this exception: when converting to a VARYCHAR-type destination such as the DB2 VARCHAR, DEST-MAXLEN does not include the length of the "LL" length specification. DEST-MAXLEN = 35 when converting to numeric or Sybase-decimal.
DEST-SCALE	Only when converting to or from numeric, Sybase-decimal, or packed decimal datatypes.	Number of digits that follow the decimal point in the destination variable. DEST-SCALE must be less than or equal to DEST-PRECIS and cannot be greater than 31. Use the same value as in SRC-SCALE
DEST-PRECIS	Only when converting to or from numeric, Sybase-decimal, or packed decimal datatypes.	The total number of digits in the destination data. DEST-PRECIS must be greater than or equal to DEST-SCALE and cannot be less than 1 or greater than 31. Use the same value as in SRC-PRECIS
DEST-STATUS	Not used (CS-FMT-UNUSED).	Not applicable.

Field	When used	Value represents
DEST-COUNT	Not used (CS-FMT-UNUSED).	Not applicable.
DEST-UTYPE	Not used (CS-FMT-UNUSED).	Not applicable.
DEST-LOCALE	Not used (CS-FMT-UNUSED).	Not applicable.

DESTDATA
Name of the variable that contains the converted data. This is the variable described in the *DESTDATA* structure.

OUTLEN
(O) Actual length, in bytes, of the data placed in *DESTDATA*. If the conversion fails, CSBCONVERT sets *OUTLEN* to CS-UNUSED.

Return value

CSBCONVERT returns one of the following values listed in Table 2-22.

Table 2-22: CSBCONVERT return values

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.
CS-FAIL (-2)	The routine failed.
TDS-INVALID-DATAFMT-VALUE (-181)	A SRCFMT or DESTFMT field contains an illegal value—probably an illegal datatype value.
TDS-INVALID-VAR-ADDRESS (-175)	This value cannot be NULL.
TDS-MONEY-CONVERSION-ERROR (-22)	Converting TDSMONEY4 failed, possibly because the TDS version is not 4.2 or above.
TDS-INVALID-DATA-CONVERSION (-172)	This value cannot be NULL.
TDS-INVALID-LENGTH(-173)	Converting TDSMONEY4 failed, possibly because the TDS version is not 4.2 or above.

Examples

The following code fragment demonstrates the use of CSBCONVERT. It is taken from the sample program SYCTSAA5 in Appendix A, “Sample Language Requests.”

```
*=========================================================
*==                                                     ==
*== Subroutine to fetch row processing                  ==
*==                                                     ==
*=========================================================
 FETCH-ROW-PROCESSING.
     CALL 'CTBFETCH' USING CSL-CMD-HANDLE,
                           CSL-RC,
                           CS-UNUSED,
                           CS-UNUSED,
```

```
                                CS-UNUSED,
                                FF-ROWS-READ.
           EVALUATE CSL-RC
               WHEN CS-SUCCEED
                    MOVE 'Y'              TO SW-FETCH
                    MOVE CS-VARCHAR-TYPE TO DF-DATATYPE
                    MOVE LENGTH OF CF-COL-FIRSTNME-TXT
                                          TO DF-MAXLENGTH
                    MOVE CS-CHAR-TYPE     TO DF2-DATATYPE
                    MOVE LENGTH OF CF-COL-FIRSTNME-CHAR
                                          TO DF2-MAXLENGTH
                    CALL 'CSBCONVE' USING CSL-CTX-HANDLE,
                                          CSL-RC,
                                          DATAFMT,
                                          CF-COL-FIRSTNME,
                                          DATAFMT2,
                                          CF-COL-FIRSTNME-CHAR,
                                          CF-COL-LEN
                    IF CSL-RC NOT EQUAL CS-SUCCEED
                      THEN
                         MOVE SPACES TO MSGSTR
                         STRING 'CSBCONVERT CS-VARCHAR-TYPE failed'
                                     DELIMITED BY SIZE INTO MSGSTR
                         PERFORM PRINT-MSG
                         PERFORM ALL-DONE
                    END-IF
                    COMPUTE FF-ROW-NUM = FF-ROW-NUM + 1
           ************************************
           * save ROW RESULTS for later display *
           ************************************
```

Usage

- A client application can use this function to convert the datatype of RPC return parameters to the datatype of the target server, and to convert the datatype of a retrieved value to a datatype that can be used by Open ClientConnect. This function converts a single variable each time it executes.
- When converting columns, an application must issue a separate CSBCONVERT call for each column to be converted. If several rows of data need converting, the application must issue a separate CSBCONVERT call for every column that needs conversion in each row.
- Table 2-23 lists the conversions you can perform with CSBCONVERT.

Table 2-23: Conversions performed by CSBCONVERT

Source type	Result type	Notes
CS-VARCHAR	CS-CHAR	Does EBCDIC to ASCII conversion; pads with blanks.
CS-CHAR	CS-VARCHAR	
CS-MONEY	CS-CHAR	
CS-MONEY	CS-VARCHAR	
CS-REAL	CS-FLOAT	Truncates low order digits.
CS-MONEY	CS-FLOAT	
CS-PACKED370	CS-FLOAT	
CS-FLOAT	CS-REAL	Pads with zeroes.
CS-MONEY	CS-PACKED370	
CS-CHAR	CS-PACKED370	
CS-VARCHAR	CS-PACKED370	
CS-FLOAT	CS-PACKED370	
CS-NUMERIC	CS-PACKED370	
CS-DECIMAL	CS-PACKED370	
CS-NUMERIC	CS-CHAR	
CS-DECIMAL	CS-CHAR	
CS-DATETIME	CS-CHAR	
CS-CHAR	CS-NUMERIC	
CS-CHAR	CS-DECIMAL	
CS-PACKED370	CS-NUMERIC	
CS-PACKED370	CS-DECIMAL	
CS-PACKED370	CS-CHAR	

Warning! Converting CS-MONEY or CS-CHAR values to CS-FLOAT can result in a loss of precision. Converting a CS-FLOAT value to a character type can also result in a loss of precision.

See also

Related functions

- CTBFETCH on page 107

Related topics

- “DATAFMT structure” on page 170
- “Datatypes” on page 174

CSBCTXALLOC

Description

Allocates a context structure.

Syntax

```
COPY CTPUBLIC.

01 VERSION      PIC S9(9) COMP SYNC.
01 RETCODE      PIC S9(9) COMP SYNC.
01 CONTEXT      PIC S9(9) COMP SYNC.

CALL 'CSBCTXAL' USING VERSION RETCODE CONTEXT.
```

Parameters

VERSION

(I) Version of Client-Library behavior that the application expects. The following table lists the symbolic values that are legal for *VERSION*:

Value	Indicates	Supported features
CS-VERSION-46	Communicates with Adaptive Server release 4.6.	RPCs. **Note** This is the initial version of Client-Library.
CS-VERSION-50	Communicates with Adaptive Server release 10.0 and above.	RPCs.

RETCODE

(O) Variable where the result from an executed function returns. Its value is one of the codes listed under "Return value," in this section.

CONTEXT

(O) Variable where this newly-allocated context structure returns. This is the name used by CTBINIT when it initializes Client-Library.

Return value

CSBCTXALLOC returns one of the following values:

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.
CS-FAIL (-2)	The routine failed. The most common cause for a CSBCTXALLOC failure is a lack of available memory.

Examples

The following code fragment demonstrates the use of CSBCTXALLOC. It is taken from the sample program SYCTSAA5 in Appendix A, "Sample Language Requests."

```
*****************************
*    PROGRAM INITIALIZATION *
*****************************
     MOVE C-N    TO NO-MORE-MSGS-SW.
```

```
           MOVE C-N    TO NO-ERRORS-SW.
           MOVE C-Y    TO SW-DIAG.
           COMPUTE PAGE-CNT = PAGE-CNT + 1.
           PERFORM GET-SYSTEM-TIME.
           MOVE LOW-VALUES TO A5PANELO.
           MOVE -1         TO SERVERL.
       GET-INPUT-AGAIN.
           PERFORM DISPLAY-INITIAL-SCREEN.
           PERFORM GET-INPUT-DATA.
      **********************************
      *    ALLOCATE A CONTEXT STRUCTURE *
      **********************************
           MOVE ZERO TO CSL-CTX-HANDLE.
           CALL 'CSBCTXAL' USING CS-VERSION-50
                                 CSL-RC
                                 CSL-CTX-HANDLE.
           IF CSL-RC NOT EQUAL CS-SUCCEED
             THEN
                MOVE SPACES TO MSGSTR
                STRING 'CSBCTXAL failed' DELIMITED BY SIZE INTO MSGSTR
                PERFORM PRINT-MSG
                PERFORM ALL-DONE
           END-IF.
      **********************************
      * INTITIALIZE THE CLIENT-LIBRARY *
      **********************************
           CALL 'CTBINIT' USING CSL-CTX-HANDLE
                                CSL-RC
                                CS-VERSION-50.
           IF CSL-RC NOT EQUAL CS-SUCCEED
             THEN
                MOVE SPACES TO MSGSTR
                STRING 'CTBINIT failed' DELIMITED BY SIZE INTO MSGSTR
                PERFORM PRINT-MSG
                PERFORM ALL-DONE
           END-IF.
           PERFORM PROCESS-INPUT.
           PERFORM QUIT-CLIENT-LIBRARY.
           GOBACK.
```

Usage

- CSBCTXALLOC allocates a context structure.
- A context structure contains information that describes an application context. For example, a context structure defines the version of Client-Library that is in use.

- Allocating a context structure is the first step in any Client-Library application.
- After allocating a context structure, a Client-Library application typically customizes the context by calling CSBCONFIG and/or CTBCONFIG, then sets up one or more connections within the context.
- To deallocate a context structure, an application calls CSBCTXDROP.

See also

Related functions

- CSBCONFIG on page 148
- CTBCONALLOC on page 55
- CTBCONFIG on page 64
- CSBCTXDROP on page 162

CSBCTXDROP

Description

Deallocates a context structure.

Syntax

```
COPY CTPUBLIC.

01 CONTEXT        PIC S9(9) COMP SYNC.
01 RETCODE        PIC S9(9) COMP SYNC.

CALL 'CSBCTXDR' USING CONNECTION RETCODE.
```

Parameters

CONTEXT

(I) A context structure.

RETCODE

(O) Variable where the result of function execution is returned. Its value is one of the codes listed under “Return value,” in this section.

Return value

CSBCTXDROP returns one of the following values:

Value	Meaning
CS-SUCCEED (-1)	The routine completed successfully.
CS-FAIL (-2)	The routine failed. The most common cause for a CSBCTXDROP failure is that the context contains an open connection.

Examples

The following code fragment demonstrates the use of CSBCTXDROP at the end of a program, after results have been processed. It is taken from the sample program SYCTSAA5 in Appendix A, “Sample Language Requests.”

```
*================================================
*==                                            ==
*== Subroutine to perform exit client library and ==
*== deallocate context structure.              ==
*==                                            ==
*================================================
 QUIT-CLIENT-LIBRARY.
**************************
* EXIT THE CLIENT LIBRARY *
**************************
     CALL 'CTBEXIT' USING CSL-CTX-HANDLE
                          CSL-RC
                          CS-UNUSED.
     IF CSL-RC = CS-FAIL
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBEXIT failed' DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
     END-IF.
***********************************
* DE-ALLOCATE THE CONTEXT STRUCTURE *
***********************************
     CALL 'CSBCTXDR' USING CSL-CTX-HANDLE
                           CSL-RC.
     IF CSL-RC = CS-FAIL
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CSBCTXDR failed' DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
     END-IF.
     EXEC CICS RETURN END-EXEC.
 QUIT-CLIENT-LIBRARY-EXIT.
     EXIT.
```

Usage

- CSBCTXDROP deallocates a context structure.
- A context structure describes a particular context, or operating environment, for a set of server connections.
- Once a context has been deallocated, it cannot be reused. To allocate a new context, an application calls CSBCTXALLOC.

- A Client-Library application cannot call CSBCTXDROP to deallocate a context structure until it has called CTBEXIT to clean up Client-Library space associated with the context.
- CSBCTXDROP fails if the context contains an open connection.

See also

Related functions

- CSBCTXALLOC on page 160
- CTBCLOSE on page 37
- CTBEXIT on page 104

CHAPTER 3

Topics

The following topics are included in this chapter:

- Buffers
- CLIENTMSG structure
- Customization
- DATAFMT structure
- Datatypes
- Error and message handling
- The CS-EXTRA-INF property
- Nulls
- Properties
- Remote procedure calls (RPCs)
- Results
- SERVERMSG structure
- SQLCA structure
- SQLCODE structure
- Handles

Buffers

Description

A number of arguments used in Client-Library functions affect the contents of buffers: *ACTION*, *BUFFER*, *BUFFER-LEN*, *BUFBLANKSTRIP*, *OUTLEN*.

These arguments are described individually below. For a summary of argument values and their interaction, see Table 3-1 on page 167.

Arguments

ACTION describes what is done to the data. For most functions, *ACTION* can take the following symbolic values:

CS-SET	Assigns a value
CS-GET	Retrieves a value
CS-CLEAR	Clears the buffer or, for functions that set properties, resets to the default property value

BUFFER is a program variable, called a "buffer."

When ACTION is CS-SET	BUFFER is the data being read by the function.
When ACTION is CS-GET	BUFFER is the information retrieved by the function.
When ACTION is CS-CLEAR	BUFFER remains unchanged.

BUFFER-LEN is the length, in bytes, of the *BUFFER* argument.

When the value in the buffer is a fixed-length or symbolic value	Assign BUFFER-LEN the actual length of the value or CS-UNUSED.
When the value in the buffer is of variable length	Assign BUFFER-LEN the maximum number of bytes that the buffer can contain.

If *BUFFER-LEN* is set to CS-GET and the buffer is too small to hold the returned value, the function returns CS-FAIL in the *RETCODE* argument and the actual length of the requested information in *OUTLEN*.
When this happens, you can query the length of the incoming data by setting *BUFFER-LEN* to 0 and executing the function. When this is done, the actual length of the data is returned to the *OUTLEN* argument. Enlarge the buffer, assign the value in *OUTLEN* to *BUFFER-LEN*, and execute the function again.
BUFBLANKSTRIP is the blank stripping indicator. It indicates whether or not trailing blanks are stripped. This argument is assigned one of the following symbolic values:

CS-TRUE	Trailing blanks are stripped. The value in the buffer ends at the last non-blank character it contains.
CS-FALSE	Trailing blanks are not stripped; they are included in the value.

OUTLEN is an integer variable where the function returns the actual length, in bytes, of the data being retrieved during a CS-GET operation. For all other operations, *OUTLEN* is ignored.

When the retrieved data is longer than *BUFFER-LEN* bytes, an application can use the value of *OUTLEN* to determine how many bytes are needed to hold the information.
To do this, set the value of *OUTLEN* to 0 and call the function. Then set the value of *BUFFER-LEN* to the length returned in *OUTLEN* and call the function again.

Summary of buffer entries

Table 3-1 summarizes the interaction between *ACTION*, *BUFFER*, *BUFFER-LEN*, and *OUTLEN*.

Table 3-1: ACTION, BUFFER, BUFFER-LEN, and OUTLEN

For this ACTION	When BUFFER	BUFFER-LEN is	OUTLEN is	Result
CS-CLEAR	N/A	CS-UNUSED	Ignored	The property reverts to its default value.
CS-SET	Contains a variable-length character string	The length of the string	Ignored	The data in *BUFFER* is read by the function.
CS-SET	Contains a variable-length character string for which the terminating character is the last non-blank character	The maximum length of the string	Ignored	The application sets the value of *BUFBLANKSTRIP* to CS-TRUE. The data in *BUFFER* is read by the function.
CS-SET	Contains a fixed-length character string or symbolic value	CS-UNUSED	Ignored	The data in *BUFFER* is read by the function.
CS-GET	Is large enough for the return character string	The length of the buffer	Set by Client-Library	The retrieved value is copied to the buffer. *OUTLEN* returns the length of the retrieved value.
CS-GET	Is not large enough for the return character string	The length of the buffer	Set by Client-Library	No data is copied to the buffer. *OUTLEN* returns the length of the value being retrieved. The function returns CS-FAIL, indicating that you should assign the value returned here to the length argument and execute the program again.

For this ACTION	When BUFFER	BUFFER-LEN is	OUTLEN is	Result
CS-GET	Is assumed to be large enough for a fixed-length or symbolic value	CS-UNUSED	Set by Client-Library	The retrieved value is copied to the buffer. *OUTLEN* returns the length of the value being retrieved.

CLIENTMSG structure

Description

A CLIENTMSG (client message) structure contains information about an error or informational message returned by Open ClientConnect. This structure is defined within the application. When the error involves interaction with the operating system, the operating system error information is returned to this structure. CTBDIAG returns a message string and information about the message into CLIENTMSG.

Server messages are returned to a SERVERMSG structure, described in "SERVERMSG structure" on page 194. A sample CLIENTMSG structure is provided in the CTPUBLIC copybook.

A CLIENTMSG structure is defined as follows:

```
01 CLIENTMSG.
05 CMSG-SEVERITY PIC S9(9) COMP SYNC.
05 CMSG-OC-MSGNO PIC S9(9) COMP SYNC.
05 CMSG-OC-MSGTEXT PIC X(256).
05 CMSG-OC-MSGTEXT-LEN PIC S9(9) COMP SYNC.
05   CMSG-OS-MSGNO PIC S9(9) COMP SYNC.
05 CMSG-OS-MSGTEXT PIC X(256).
05 CMSG-OS-MSGTEXT-LEN PIC S9(9) COMP SYNC.
05 CMSG-STATUS PIC S9(9) COMP SYNC.
```

CMSG-SEVERITY is a symbolic value representing the severity of the message. Severity values are provided in the CTPUBLIC copybook.

Table 3-2 on page 168 lists the legal values for CMSG-SEVERITY.

Table 3-2: Values for the CLIENTMSG CMSG-SEVERITY field

CMSG-SEVERITY value	Meaning
CS-SV-INFORM (0)	No error occurred. The message is informational.
CS-SV-API-FAIL (1)	A Client-Library routine generated an error. This error is typically caused by a bad parameter or calling sequence. The server connection is probably salvageable.

CMSG-SEVERITY value	Meaning
CS-SV-RETRY-FAIL (2)	An operation failed, but the operation can be retried.
CS-SV-RESOURCE-FAIL (3)	A resource error occurred. This error is typically caused by an allocation error, a lack of file descriptors, or timeout error. The server connection is probably not salvageable.
CS-SV-CONFIG-FAIL (4)	A configuration error occurred.
CS-SV-COMM-FAIL (5)	An unrecoverable error in the server communication channel occurred. The server connection is not salvageable.
CS-SV-INTERNAL-FAIL (6)	An internal Client-Library error occurred.
CS-SV-FATAL (7)	A serious error occurred. All server connections are unusable.

- *CMSG-OC-MSGNO* is the Client-Library message number. Client-Library messages are listed in the Mainframe Connect Client Option and Server Option *Messages and Codes*.
- *CMSG-OC-MSGTEXT* is the text of the Client-Library message string.
- *CMSG-OC-MSGTEXT-LEN* is the length, in bytes, of *CMSG-OC-MSGTEXT*. If there is no message text, the value of *CMSG-OC-MSGTEXT-LEN* is 0.
- *CMSG-OS-MSGNO* is the server error number, if any. A value here indicates that the message involved CICS or IMS TM I/O errors, remote server errors, or Transaction Router Service (TRS) errors.
- *CMSG-OS-MSGTEXT* is the text of the operating system message string, if any.
- *CMSG-OS-MSGTEXT-LEN* is the length, in bytes, of *CMSG-OS-MSGTEXT*. If there is no message text, the value of *CMSG-OS-MSGTEXT-LEN* is 0.
- *CMSG-STATUS* is reserved for future use.

Customization

Description

When installing Open ClientConnect, system programmers customize the product for the customer site, defining language and program characteristics locally. Some of the customized items are used by Gateway-Library programs.

The following locally-defined items are used by Gateway-Library functions:

- An access code, which is required to retrieve a client's password.

  Two customization options are related to the ability to retrieve client passwords:

  - The access code itself is defined during customization.
  - An access code flag is set to indicate whether the access code is required to retrieve the client password.
- The native language used at the mainframe (The default is US-English).
- Whether DB2 LONG VARCHAR data strings with lengths that are greater than 255 bytes are truncated or rejected when sent to a client.

for customization instructions, see the Mainframe Connect Server Option *Installation and Administration Guide*. The customization module is loaded during program initialization (CTBINIT).

DATAFMT structure

Description

A DATAFMT structure is used to describe data values and program variables. For example:

- CTBBIND requires a DATAFMT structure describing a destination variable.
- CTBDESCRIBE returns a DATAFMT structure describing a result data item.
- CTBPARAM requires a DATAFMT structure describing an input parameter.
- CSBCONVERT requires a DATAFMT structure describing source and destination data.

Most functions use only a subset of the fields in a DATAFMT structure. For example, CTBBIND does not use the FMT-NAME, FMT-STATUS, and FMT-UTYPE fields, and CTBDESCRIBE does not use the FMT-FORMAT field. For information on which DATAFMT fields a function uses, see Table 3-3 in this chapter, or descriptions of the CTBBIND and CTBDESCRIBE functions located in the sections CTBBIND on page 25 and CTBDESCRIBE on page 78 in Chapter 2, "Functions."

A DATAFMT structure is defined as follows:

```
01 DATAFMT
05 FMT-NAME PIC X(132).
05 FMT-NAMELEN PIC S9(9) COMP SYNC.
05 FMT-TYPE PIC S9(9) COMP SYNC.
05 FMT-FORMAT PIC S9(9) COMP SYNC.
05 FMT-MAXLEN PIC S9(9) COMP SYNC.
05 FMT-SCALE PIC S9(9) COMP SYNC.
05 FMT-PRECIS PIC S9(9) COMP SYNC.
05 FMT-STATUS PIC S9(9) COMP SYNC.
05 FMT-COUNT PIC S9(9) COMP SYNC.
05 FMT-UTYPE PIC S9(9) COMP SYNC.
05 FMT-LOCALE PIC S9(9) COMP SYNC.
```

Table 3-3 describes the fields in the DATAFMT structure.

Table 3-3: Fields in the DATAFMT structure

Field	Contents	Used by
FMT-NAME	The name of the data item.	CTBDESCRIBE CTBPARAM
FMT-NAMELEN	The length of FMT-NAME.	CTBDESCRIBE CTBPARAM
FMT-TYPE	The datatype of the data. See the specific call to find which data this refers to.	CSBCONVERT CTBBIND CTBDESCRIBE CTBPARAM
FMT-FORMAT	The format of the data, represented by symbolic values.	CTBBIND
FMT-MAXLEN	The maximum length of the data.	CSBCONVERT CTBBIND CTBDESCRIBE CTBPARAM
FMT-SCALE	The number of digits in the decimal part of a number. This field is used with packed decimal, numeric and Sybase-decimal.	CSBCONVERT CTBBIND
FMT-PRECIS	The total number of digits in a number. This field is used with packed decimal, numeric and Sybase-decimal.	CSBCONVERT CTBBIND
FMT-STATUS	Status values.	CTBDESCRIBE CTBPARAM
FMT-COUNT	The number of items.	CTBBIND CTBDESCRIBE
FM-UTYPE	The user-defined datatype (UDT) of retrieved data. The UDT is assigned by the server.	CTBDESCRIBE CTBPARAM

Field	Contents	Used by
FMT-LOCALE	Reserved for future use.	CSBCONVERT CTBBIND CTBDESCRIBE CTBPARAM

- FMT-NAME is the name of the data item. This can be a column, a parameter, or a return status name.
- FMT-NAMELEN is the length, in bytes, of FMT-NAME. Assign FMT-NAMELEN a value of 0 if the data item is unnamed.
- FMT-TYPE is the datatype of the data. This is one of the Client-Library datatypes listed under "Datatypes" on page 174.

Note Return status values have a datatype of CS-INT.

- FMT-FORMAT is the destination format of fixed-length character or binary data. FMT-FORMAT can have the following values listed in Table 3-4.

Table 3-4: Values for the DATAFMT field FMT-FORMAT

Value	Meaning	Datatype
CS-FMT-PADBLANK	The data should be padded with blanks to the full length of the destination variable.	For fixed-length character data only.
CS-FMT-PADNULL	The data should be padded with LOW-VALUES to the full length of the destination variable.	For binary or fixed-length character data.

- FMT-MAXLEN can represent various lengths, depending on which function is using the DATAFMT structure. Lengths are represented in bytes. Table 3-5 on page 172 lists the meaning of FMT-MAXLEN for each function that uses it.

Table 3-5: Lengths defined by the DATAFMT field FMT-MAXLEN

Function	Length defined
CSBCONVERT	The length of the source variable and the length of the destination variable.
CTBBIND	The length of the variable to which the data is bound.
CTBDESCRIBE	The maximum possible length of the column or parameter being described.
CTBPARAM	The maximum length of return parameter data.

- FMT-SCALE is the number of decimal places in the value being converted. FMT-SCALE is used with CTBBIND, CSBCONVERT, and CTBPARAM when converting to or from decimal datatypes CS-PACKED370, numeric, and Sybase-decimal.

- Legal values for FMT-SCALE are from 0 to 31. If the actual scale is greater than 31, the call fails.
- To indicate that destination data should use the same scale as the source data, set FMT-SCALE to CS-SRC-VALUE. FMT-SCALE must be less than or equal to FMT-PRECIS.

- FMT-PRECIS is the precision of the value being converted. FMT-PRECIS is used only with CTBBIND, CSBCONVERT, and CTBPARAM when converting to or from decimal datatypes CS-PACKED370, numeric, and Sybase-decimal.
  - Legal values for FMT-PRECIS are from 1 to 31. To indicate that destination data should use the same precision as the source data, set FMT-PRECIS to CS-SRC-VALUE. The value of FMT-PRECIS must be greater than or equal to FMT-SCALE.
- FMT-STATUS is one or more of the following symbolic values (added together) listed in Table 3-6.

Table 3-6: Values for the DATAFMT field FMT-STATUS

Value	Meaning	For this function
CS-CANBENULL	The column can contain nulls.	CTBDESCRIBE
CS-NODATA	No data is associated with the result data item.	CTBDESCRIBE
CS-INPUTVALUE	The parameter is a non-return RPC parameter (input parameter).	CTBPARAM
CS-RETURN	The parameter is an RPC return parameter.	CTBPARAM

- FMT-COUNT is the number of rows to copy to program variables per CTBFETCH call.
- FMT-UTYPE is the user-defined datatype, if any, of data returned by the server.

Note FMT-UTYPE is used only for datatypes defined at the server, not for datatypes defined by Open ClientConnect. For example, these would include Adaptive Server-defined datatypes or datatypes defined with TDSETUDT in Open ServerConnect.

- FMT-LOCALE is reserved for future use. It must be set to zero.

Datatypes

Description

Open ClientConnect supports a wide range of datatypes. These datatypes are shared with Open Client, Open Server and Open ServerConnect, and correspond directly to Adaptive Server datatypes.

Table 3-7 lists the Client-Library datatypes, together with the corresponding type constants, Adaptive Server datatypes, and Open ServerConnect datatypes.

Table 3-7: Summary of Open ClientConnect datatypes

This Client-Library datatype	Whose datatype declaration looks like this	Describes this type of data	Corresponds to this Adaptive server datatype	Corresponds to this Open ServerConnect datatype
CS-BINARY	PIC X(n)	Binary	binary	TDSBINARY
CS-CHAR	PIC X(n)	Character	char	TDSCHAR
CS-DATETIME	01 MY-TIME 49 DATE PIC S9(9) 49 TIME PIC S9(9)	8-byte datetime	datetime	TDSDATETIME
CS-DATETIME4	01 MY-TIME 49 DATE PIC S9(4) 49 TIME PIC S9(4)	4-byte datetime	smalldatetime	TDSDATETIME4
CS-FLOAT	USAGE COMP-2	8-byte float	float	TDSFLT8
CS-INT	PIC S9(5-9)	4-byte integer	int	TDSINT4
CS-LONGBINARY	PIC X(n)	Long variable binary	--	TDSLONGVARBIN
CS-LONGCHAR	PIC X(n)	Long variable character	--	TDSLONGVARCHAR
CS-MONEY	01 MY-MONEY 49 HIGH PIC S9(9) 49 LOW PIC S9(9)	8-byte money	money	TDSMONEY
CS-MONEY4	PIC S9(9) COMP	4-byte money	smallmoney	TDSMONEY4
CS-PACKED370	PIC S9(n) V9(m) USAGE COMP-3	IBM S/370 packed decimal	decimal	TDS-PACKED-DECIMAL
CS-REAL	USAGE COMP-1	4-byte float	real	TDSFLT4
CS-SMALLINT	PIC S9(3-4)	2-byte integer	smallint	TDSINT2

This Client-Library datatype	Whose datatype declaration looks like this	Describes this type of data	Corresponds to this Adaptive server datatype	Corresponds to this Open ServerConnect datatype
CS-VARBINARY	01 MY-VARBINARY 49 LEN PIC S9(4) 49 ARR PIC X(n)	Variable-length binary	--	TDSVARYBIN
CS-VARCHAR	01 MY-VARCHAR 49 LEN PIC S9(4) 49 ARR PIC X(n)	Variable-length character	--	TDSVARYCHAR
CS-NUMERIC	PIC X(35)	--	numeric	TDSNUMERIC
CS-DECIMAL	PIC X(35)	--	Sybase decimal	TDS-SYBASE-DECIMAL

Open ClientConnect datatypes

Open ClientConnect datatypes are designed to match the corresponding DB2 datatypes. For example, CS-VARCHAR is the same as the DB2 datatype VARCHAR, which is different from the Adaptive Server varchar type.

Binary

Open ClientConnect supports the following three binary types:

- CS-BINARY is a binary type.
- CS-VARBINARY is a variable-length binary type. It corresponds to the DB2 datatype VARBINARY, the DB-Library datatype DBVARYBIN, and the Gateway-Library datatype TDSVARYBIN, and includes a length specification (the initial two bytes, referred to in print as “LL”) along with the data.
- CS-LONGBINARY is a long variable binary type. It does not include the two-byte “LL” length specification prefix. The default maximum length for this datatype is 32K.

Note CS-VARBINARY does not correspond to the Adaptive Server datatype varbinary. Open ClientConnect converts Adaptive Server varbinary data to CS-VARBINARY.

Character

Open ClientConnect supports the following three character types:

- CS-CHAR is a set-length character type.

- CS-VARCHAR is a variable-length character type. It corresponds to the DB2 datatype VARCHAR, the DB-Library datatype DBVARYCHAR, and the Gateway-Library datatype TDSVARYCHAR, and includes a length specification (the initial two bytes, referred to in print as “LL”) along with the data.
- CS-LONGCHAR is a long variable-length character type. It does not include the two-byte “LL” length specification prefix.

Note CS-VARCHAR does not correspond to the Adaptive Server datatype varchar. Open ClientConnect converts Adaptive Server varchar data to CS-VARCHAR.

Datetime

Open ClientConnect supports the following two datetime types that hold 8-byte and 4-byte datetime values, respectively:

- CS_DATETIME corresponds to the Adaptive Server datatype datetime. The range of legal CS_DATETIME values is from January 1, 1753 to December 31, 9999, with a precision of 1/300th of a second (3.33 milliseconds).
- CS_DATETIME4 corresponds to the Adaptive Server datatype smalldatetime. The range of legal CS-DATETIME4 values is from January 1, 1900 to June 6, 2079, with a precision of 1 minute.

Integer

Open ClientConnect supports the following two integer types:

- CS-SMALLINT is a 2-byte integer.
- CS-INT is a 4-byte integer.

Real, float, packed decimal, numeric and Sybase-decimal

Open ClientConnect supports the following five decimal types:

- CS-REAL is a 4-byte decimal value.
- CS-FLOAT is an 8-byte decimal value.
- CS-PACKED370 is used to handle IBM S/370 packed decimal data. It can be converted to the Adaptive Server money, char, numeric, or Sybase-decimal datatype with CSBCONVERT.

  CS-PACKED370 values can be negative. The maximum number of decimal places for a packed decimal object is 31.
- CS-NUMERIC is used to handle Adaptive Server numeric data. It can be converted to character or packed decimal.

```
01 NUMDEC
05 Precision CHAR(1)
05 Scale     CHAR(1)
05 ARR       CHAR(33)
```

- CS-DECIMAL is used to handle Adaptive Server numeric data. It can be converted to character or packed decimal. It is defined the same way as CS-NUMERIC.

Money

Open ClientConnect supports the following two money datatypes that hold 8-byte and 4-byte money values, respectively:

- CS-MONEY corresponds to the Adaptive Server datatype money. The range of legal CS-MONEY values is +/- $922,337,203,685,477.5807.
- CS-MONEY4 corresponds to the Adaptive Server datatype smallmoney. The range of legal CS-MONEY4 values is between -$214,748.3648 and +$214,748.3647.

Error and message handling

Description

All Open ClientConnect functions return success or failure indications in their *RETCODE* arguments. It is highly recommended that applications check these return codes for each call.

In addition, Open ClientConnect applications must handle three types of error and informational messages:

- Open ClientConnect messages, also known as “client messages,” are generated by the functions documented in this book. They range in severity from informational messages to fatal errors.
- Operating system messages.
- Server messages—messages generated by the server. Server messages also range in severity from informational messages to fatal errors.

For a list of messages, see the Mainframe Connect Client Option and Server Option *Messages and Codes.*

Return codes

Client-Library return codes begin with “CS-”. The codes returned to each Client-Library function are listed on the reference pages for that function, under “Returns.”

Gateway-Library return codes all begin with “TDS-”. However, on the mainframe, a few TDS-XXX return codes are returned to Client-Library functions. Some are returned to both Gateway-Library and Client-Library functions.

TDS-XXX return codes that are returned to specific functions only are documented on the reference pages for those functions, under “Returns.” Programs using these functions should check for these codes.

Some TDS-XXX return codes can be returned to any function under certain circumstances.

You can find a list of all TDS-XXX codes in the Mainframe Connect Client Option and Server Option *Messages and Codes*.

Messages

A Client-Library application uses the function CTBDIAG to handle messages in line. CTBDIAG returns message information to two structures defined within the application: the CLIENTMSG structure for client messages, and the SERVERMSG structure for server messages.

An application calls CTBDIAG to initialize in-line message handling for a connection. CTBDIAG cannot be used at the context level.

Note Whenever a Client-Library function returns CS-FAIL, you must run CTBDIAG to determine what the error is.

An application can retrieve messages into SQLCA and SQLCODE structures. If the application uses these structures, it must set the property CS-EXTRA-INF to CS-TRUE, using CTBCONALLOC. This is because the SQL structures require information that Client-Library does not customarily return. If CS-EXTRA-INF is not set, a loss of information occurs. For more information, see “The CS-EXTRA-INF property” on page 179, and “SQLCA structure” on page 197.

For additional information on the in-line method of handling function and server messages, see CTBDIAG on page 86 and the topics “CLIENTMSG structure” on page 168 and “SERVERMSG structure” on page 194.

The CS-EXTRA-INF property

The CS-EXTRA-INF property is used by an application to determine the number of rows affected by the most recent command. If you want this extra information, you must set this property before you call the function.

An application can determine the number of rows in two ways:

- An application that is retrieving messages into a SQLCA or SQLCODE structure must set the property CS-EXTRA-INF to CS-TRUE, using CTBCONPROPS. This is necessary because the SQLCA structure needs to know the number of rows affected (information that functions do not customarily return). If CS-EXTRA-INF is not set, a loss of information occurs.
- An application that is not using SQLCA or SQLCODE can also set CS-EXTRA-INF to CS-TRUE. In this case, the extra information is returned as standard Client-Library messages.

For further information on the use of the SQLCA and SQLCODE structures in Open ClientConnect, turn to "SQLCA structure" on page 197 and "SQLCODE structure" on page 198.

For three-tier processing, requests directed to a Adaptive Server or an Open Server application are sent to Mainframe ClientConnect (MCC), supplied with the Sybase DirectConnect product.

For two-tier processing, requests are sent directly to an Adaptive Server.

MCC allows mainframe transactions to access the LAN-based environment in which Sybase servers operate. It logs into the target server, passing along login information, does the necessary conversions, and forwards the request. When results are ready, it passes them from the server to the client, again performing any necessary conversions.

MCC is transparent to the mainframe application. You specify the name of the server in the SERVERNAME parameter of the CTBCONNECT call, and the MCC forwards the request to the specified server.

To learn how CICS or IMS TM determines the MCC and connections to use to connect to the target server, see the Mainframe Connect Server Option *Installation and Administration Guide*. The client program is not concerned with these details.

Note Requests to Open ServerConnect are sent directly from one transaction processing station to another and do not use Mainframe ClientConnect.

Nulls

Description

Client-Library allows parameters to have a “null” value; that is, to contain no information, not even blanks. A null value is usually represented in COBOL by LOW-VALUES.

NULL and unused parameters

There are several rules for assigning null values to arguments:

- For *handles*, an application assigns the following symbolic values to indicate a null:
  - CS-NULL-CONTEXT for context handles.
  - CS-NULL-CONHANDLE for connection handles
  - CS-NULL-CMD for command handles
- For *output arguments*:
  - Arguments that return a single integer are never null.
  - Arguments that return longer values can be null. A null value is indicated by a separate argument, indicating that the argument of interest should be treated as null.
- For *arguments that have a corresponding length argument*, assign the value CS-NULL-STRING to the corresponding length argument, if one is present, to indicate that the value of an argument should be treated as null.

Note For DATAFMT structures, you indicate a null field by setting FMT-MAXLEN to zero.

- For *arguments that have NULL indicators*, assign CS-PARAM-NULL to the indicator argument.
  - For example, CTBBIND has the arguments *COPIED* and *COPIED-NULL*. If *COPIED* is null, assign CS-PARAM-NULL to *COPIED-NULL*. If COPIED is not null, assign CS-PARAM-NOTNULL to *COPIED-NULL*.
- For *all other variables*, assign CS-UNUSED to indicate that the argument should be ignored.

Padding with NULLs

The FMT-FORMAT field of the DATAFMT structure of CTBBIND can be padded with either blanks or nulls. CS-FMT-PADNULL pads with LOW-VALUES; CS-FMT-PADBLANK pads with blanks.

Properties

Description

Properties define aspects of Open ClientConnect behavior.

login properties

Login properties are used when logging into a server. Login properties include CS-USERNAME, CS-PASSWORD, CS-PACKETSIZE, and CS-NET-DRIVER (used with dynamic network drivers).

Negotiated properties

A server can change the values of some login properties during the login process. For example, if an application sets CS-PACKETSIZE to 2048 bytes and then logs into a server that cannot support this packet size, the server overwrites 2048 with a packet size it can support. These types of properties are called *negotiated properties*.

CS-NET-DRIVER is used to switch communication protocols. The supported protocols vary depending on the operating environment. TCP/IP and CPI-C are supported in CICS, IMS and MVS. SNA (LU6.2) is only supported in CICS and IMS.

Setting and retrieving properties

An application calls CTBCONFIG, CTBCONPROPS, and CTBCMDPROPS to set and retrieve properties at the context, connection, and command structure levels, respectively. An application calls CTBCONFIG to set and retrieve most context properties; it calls CSBCONFIG to set and retrieve global context properties.

When a context structure is allocated, its property values default to standard values.

When a connection structure is allocated, it picks up default property values from its parent context. For example, if CS-TEXTLIMIT is set to 16,000 at the context level, then any connection created within this context has a default text limit value of 16,000. Likewise, when a command structure is allocated, it picks up default property values from its parent connection.

An application can override a default property value by calling CSBCONFIG, CTBCONFIG, CTBCONPROPS, or CTBCMDPROPS to change the value of the property.

Most property values can be either set or retrieved by an application, but some properties are “retrieve only.”

Summary of properties

Table 3-8 lists the Open ClientConnect properties.

Table 3-8: Open ClientConnect properties

Property	Meaning	Values	Function set by	Notes
CS-APPNAME	The application name used when logging into the server.	A character string. The default is NULL.	CTBCONPROPS	Login property. Takes effect only if set before the connection is established.
CS-CHARSETCNV	The conversion indicator. It indicates whether or not character set conversion is taking place.	CS-TRUE or CS-FALSE. A default is not applicable.	CTBCONPROPS	Retrieve only, after the connection is established.
CS-COMMBLOCK	A pointer to a communication sessions block (EIB).	A pointer value. The default is NULL.	CTBCONPROPS	Takes effect only if set before the connection is established.
CS-EXTRA-INF	The extra information indicator. It specifies whether or not to return the extra information that is required when processing messages in-line using a SQLCA or SQLCODE.	CS-TRUE or CS-FALSE. The default is CS-FALSE.	CSBCONFIG CTBCONFIG CTBCONPROPS	
CS-HOSTNAME	The host (server) machine name.	A character string. The default is NULL.	CTBCONPROPS	Login property. Takes effect only if set before the connection is established.
CS-LOC-PROP	A pointer to a CS-LOCALE structure that defines localization information.	A pointer value. A connection picks up a default CS-LOC-PROP from its parent context.	CTBCONPROPS	Login property.

Property	Meaning	Values	Function set by	Notes
CS-LOGIN-TIMEOUT	The login timeout value.	An integer value. The default is 60 seconds. A value of CS-NO-LIMIT represents an infinite timeout period.	CTBCONFIG	Open ClientConnect ignores this property. This is the same value as the CICS RTIMEOUT.
CS-MAX-CONNECT	The maximum number of connections for this context.	An integer value. The default varies by platform. On mainframes, the default is 25 (an unlimited number of connections can be defined for a context).	CTBCONFIG	
CS-NET-DRIVER	The type of network driver in use.	CS-LU62, CS-TCPIP, CS-INTERLINK, or CS-NCPIC. Defaults for: CICS: CS-LU62 IMS: CS-LU62 MVS: CS-NCPIC	CTBCONPROPS	
CS-NETIO	The sync/async indicator. It indicates whether network I/O is synchronous or asynchronous.	CS-SYNC-IO or CS-ASYNC-IO. The default is CS-SYNC-IO.	CTBCONFIG CTBCONPROPS	With Open ServerConnect this value is always CS-SYNC-IO.
CS-NOINTERRUPT	The interrupt indicator. It indicates whether or not the application can be interrupted.	CS-TRUE or CS-FALSE. The default is CS-FALSE, which means the application can be interrupted.	CTBCONFIG CTBCONPROPS	N/A for CICS. This property is included for compatibility with other Open Client libraries.

Property	Meaning	Values	Function set by	Notes
CS-PACKETSIZE	The TDS packet size.	An integer value. The default varies by platform. On UNIX and MVS platforms, the default is 512 bytes.	CTBCONPROPS	Negotiated login property. Takes effect only if set before the connection is established.
CS-PASSWORD	The password used to log into the server.	A character string. The default is NULL.	CTBCONPROPS	Login property. Takes effect only if set before the connection is established.
CS-TDS-VERSION	The version of the TDS protocol that the connection is using.	A symbolic version level. CS-TDS-VERSION defaults to the value of CS-VERSION.	CTBCONPROPS	Negotiated login property. Takes effect only if set before the connection is established.
CS-TEXTLIMIT	The largest text or image value to be returned on this connection.	An integer value. The default is CS-NO-LIMIT.	CTBCONFIG CTBCONPROPS	
CS-TIMEOUT	The timeout value.	An integer value. The default is CS-NO-LIMIT.	CTBCONFIG	Not supported under CICS. CICS waits for ever.
CS-TRANSACTION-NAME	A transaction name.	A string value. The default is NULL.	CTBCONPROPS	
CS-USERDATA	User-allocated data.	User-allocated data.	CTBCONPROPS CTBCMDPROPS	These are pointers to data that allow the customer to tie into the data.
CS-USERNAME	The name used to log into the server.	A character string. The default is NULL.	CTBCONPROPS	Login property. Takes effect only if set before the connection is established.

Property	Meaning	Values	Function set by	Notes
CS-VERSION	The version of Client-Library used by this context.	CS-VERSION gets its value from a context CTBINIT call.	CSBCONFIG CTBCONFIG	

About the properties

Application name

- CS-APPNAME defines the application name that a connection uses when connecting to a server.

Character set conversion

- CS-CHARSETCNV indicates whether or not the server is converting between the client and server character sets. This property is retrieve-only, after a connection is established.

  A value of CS-TRUE indicates that the server is converting between the client and server character sets; CS-FALSE indicates that no conversion is taking place.

Communications session block

- CS-COMMBLOCK defines a pointer to a communications block (EIB).

Extra information

- CS-EXTRA-INF determines whether or not Open ClientConnect returns the extra information that CTBDIAG requires to fill in SQLCA or SQLCODE structures.

  This extra information includes the number of rows affected by the most recent command.

  If an application is not retrieving messages into a SQLCA or SQLCODE, the extra information is returned as ordinary Client-Library messages.

Host name

- CS-HOSTNAME is the name of the host machine, used when logging into a server.

Locale information

- CS-LOC-PROP defines a pointer to a CS-LOCALE structure, which contains localization information. Localization information includes a language, a character set, datetime, money, and numeric formats, and a collating sequence. This property must be set to 0.

Login status

- CS-LOGIN-STATUS is CS-TRUE if a connection is open, CS-FALSE if it is not. This property can only be retrieved.
- CTBCONNECT is used to open a connection.

Login timeout

- CS-LOGIN-TIMEOUT defines the length of time, in seconds, that an application waits for a login response when making a connection attempt. Timeouts are not supported under CICS.

Maximum number of connections

- CS-MAX-CONNECT defines the maximum number of simultaneously open connections that a context can have. The default varies by platform. Negative and zero values are not allowed for CS-MAX-CONNECT.
  - On mainframes, CS-MAX-CONNECT has a default value of 25 (an unlimited number of connections can be defined for a context).
  - If CTBCONFIG is called to set a value for CS-MAX-CONNECT that is less than the number of currently open connections, CTBCONFIG generates an error and returns CS-FAIL without altering the value of CS-MAX-CONNECT.

Network driver

- CS-NET-DRIVER determines the type of dynamic network driver that is used. Possible values are:

- CS-INTERLINK
- CS-LU62
- CS-NCPIC
- CS-TCPIP

The default value for CICS and IMS is CS-LU62. The default value for MVS is CS-NCPIC.

Network I/O

- CS-NETIO determines whether a connection is synchronous or asynchronous.

  Because Open ClientConnect does not support asynchronous processing, this value is always CS-SYNC-IO.

No interrupt

- CS-NOINTERRUPT is not supported for Open ClientConnect. It is included for compatibility with other Open Client libraries.

Packet size

- CS-PACKETSIZE determines the packet size that Open ClientConnect uses when sending Tabular Data Stream (TDS) packets.
- If an application needs to send or receive large amounts of text, image, or bulk data, a larger packet size can improve efficiency. The default packet size is 512.

Password

- CS-PASSWORD defines the password that a connection uses when logging into a server.

TDS version

- CS-TDS-VERSION defines the version of the Tabular Data Stream (TDS) protocol that the connection is using.

Because CS-TDS-VERSION is a negotiated login property, its value can change during the login process. An application can set CS-TDS-VERSION to request a TDS level before calling CTBCONNECT. In this case, when CTBCONNECT creates the connection, it looks for the requested TDS version. If the server cannot provide the requested TDS version, a new (lower) TDS version is negotiated. An application can retrieve the value of CS-TDS-VERSION after a connection is established to determine the actual version of TDS in use.

The following table lists the symbolic values that CS-TDS-VERSION can have:

Value	Indicates	Features supported
CS-TDS-46	4.6 TDS	Registered procedures, TDS passthrough, negotiable TDS packet size, multi-byte character sets.
CS-TDS-50	5.0 TDS	Accesses system Adaptive Server 10.0 and above. **Note** TDS 5.0 only works with an Adaptive Server 10.0 and above.

Text and image limit

- CS-TEXTLIMIT indicates the length, in bytes, of the longest text or image value that an application wants to receive. Open ClientConnect reads but ignores any part of a text or image value that goes over this limit.
- The default value of CS-TEXTLIMIT is CS-NO-LIMIT, meaning the application reads and returns all data sent by the server.

Timeout

- CS-TIMEOUT controls the length of time, in seconds, that Client-Library waits for a server response when making a request.
- This value is ignored by Open ClientConnect.

Transaction name

- CS-TRANSACTION-NAME names a transaction. If the accessed server is a Gateway-Library application, this is the name of the transaction.
- Calls to Adaptive Server do not require a transaction name.

- All Client-Library applications can set CS-TRANSACTION-NAME. If a transaction name is not required, CS-TRANSACTION-NAME is ignored.

User data

- CS-USERDATA defines user-allocated data. This property allows an application to associate user data with a particular connection or command structure. An application allocates a data space from which it can get this data when needed.
- To associate user data with a context structure, an application calls CSBCONFIG.
- A COBOL program can use the working storage section to define this data.

User name

- CS-USERNAME defines the user login name that the connection uses to log into a server.

Version of Open ClientConnect

- CS-VERSION represents the version of Open ClientConnect behavior that an application requests through CTBINIT. The value of this property can only be retrieved.
- Connections allocated within a context pick up default CS-TDS-VERSION values from their parent context CS-VERSION level.

Remote procedure calls (RPCs)

Description

A client application can call a stored procedure on an Adaptive Server or an Open ServerConnect transaction running in a separate CICS or IMS region.

A client application can call a stored procedure or mainframe transaction in one of two ways:

- By executing a SQL language request (for example, “execute myproc”).
- By making an RPC.

Comparing RPCs and *execute* statements

RPCs have a few advantages over execute statements:

- An RPC can be used to execute an Adaptive Server stored procedure or any Open ServerConnect transaction.
- A SQL language request can only be used to execute an Adaptive Server stored procedure or a specially written Open ServerConnect language transaction.
- When sending a request to Adaptive Server, it is simpler and faster to accommodate stored procedure return parameters if the procedure is invoked with an RPC instead of a language request.

Remote Procedures executed by servers

A server can execute a stored procedure or transaction residing on another server. This might occur when a stored procedure being executed on one Adaptive Server contains an execute statement for a stored procedure on another Adaptive Server. The execute command causes the first server to log into the second server and execute the remote procedure. This is called a server-to-server RPC. It happens without any intervention from the application, although the application can specify the remote password that the first server uses to log into the second.

A server-to-server RPC also occurs when an application sends a request to execute a stored procedure that does not reside on the server to which it is directly connected.

Note SQL commands contained in a stored procedure that is executed as the result of a server-to-server remote procedure call cannot be rolled back.

RPC routines

The following functions relate to RPCs:

- CTBREMOTEPWD sets and clears the passwords that are used when logging into a remote server (This feature is not available for calls to Open ServerConnect).

- CTBCOMMAND initiates an RPC.
- CTBPARAM defines parameters for an RPC.
- CTBSEND sends an RPC.
- CTBRESULTS, CTBBIND, and CTBFETCH process remote procedure results.

RPC results

In addition to results generated by the SQL statements they contain, Adaptive Server stored procedures and Open ServerConnect transactions that are executed via an RPC:

- Can generate a return parameter result set.
- Always generate a return status result set.

All types of results—rows, status, and parameters—can be processed using CTBRESULTS, CTBBIND, and CTBFETCH.

Stored procedure return parameters

Adaptive Server stored procedures and mainframe server transactions can return values for specified "return parameters." Changes made to the value of a return parameter inside the stored procedure or transaction are then available to the program that called the procedure or transaction. This is analogous to the "pass by reference" facility available in some programming languages.

In order for a parameter to function as a return parameter, it must be declared as such within the stored procedure. Client-Library applications use the CTBPARAM routine to indicate return parameters.

Processing RPC return parameters

Return parameter values are available to an application as a parameter result set only if the application invoked the stored procedure using an RPC.

CTBRESULTS returns CS-PARAM-RESULT if a parameter result set is available to be processed. Because stored procedure parameters are returned to an application as a single row, one call to CTBFETCH copies all of the return parameters for a procedure into the program variables designated via CTBBIND. However, an application should always call CTBFETCH in a loop until it returns CS-END-DATA.

When executing a stored procedure, the server returns any parameter values immediately after returning all row results. Therefore, an application can fetch return parameters only after processing the stored procedure row results.
A stored procedure can generate several sets of row results—one for each select it contains. An application must call CTBRESULTS and CTBFETCH as many times as necessary to process these row results before calling CTBFETCH to fetch the stored procedure return parameters.

Stored procedure return status

Adaptive Server, Open Server, and Open ServerConnect applications can all return a status.

All stored procedures that run on an Adaptive Server version 4.0 or later return a status. Stored procedures usually return 0 to indicate normal completion. For a list of Adaptive Server default return status values, see return in the Adaptive Server Enterprise *Reference Manual*, which is part of the basic Sybase documentation set. Open ServerConnect status values are documented under TDSNDDON and TDSTATUS in the Mainframe Connect Server Option *Programmer's Reference for COBOL*.

Because return status values are a feature of stored procedures, only an RPC or a language request containing an execute statement can generate a return status.

When executing a stored procedure, Adaptive Server returns the status immediately after returning all other results. Therefore, an application can fetch a return status only after processing the stored procedure row and parameter results,
if any.

Open Server applications return the status after any row results, but either before or after return parameters.

Processing an RPC return status

CTBRESULTS returns CS-STATUS-RESULT if a return status result set is available to be processed. Because a return status result set contains only a single value, one call to CTBFETCH copies the status into the program variable designated via CTBBIND. However, an application should always call CTBFETCH in a loop until it returns CS-END-DATA.

Results

Description

When a client request executes a server procedure or transaction, it can generate various types of result sets that are returned to the client application. These include:

- Regular row results, which contain one or more rows of tabular data.
- Return parameter results, which contain a single row of return parameter data. Return parameters are values returned by stored procedures and transactions in the parameters (arguments) of the called function. For information on return parameters, see “Remote procedure calls (RPCs)” on page 190.
- Return status results, which consist of a single row that contains a single value, a return status. For more information on a stored procedure return status, see “Remote procedure calls (RPCs)” on page 190.

Note These are the only result types supported by Open ClientConnect. Although additional result types are supported by Open Client for other platforms, they are not supported on the mainframe.

Results are returned to an application in the form of result sets. A result set contains only a single type of result data. Regular row result sets can contain multiple rows of data, but other types of result sets contain at most a single row of data.

An application processes results by calling CTBRESULTS. The type of result available is indicated in the RESULT-TYP argument. The application calls CTBRESULTS once for each result row. CTBRESULTS returns CS-CMD-DONE in RESULT-TYP to indicate that a result set processed completely.

Some requests, for example a language request containing a Transact-SQL update statement, do not generate results. CTBRESULTS returns CS-CMD-SUCCEED to indicate the success of a request that does not return results.

SERVERMSG structure

Description

A SERVERMSG (server message) structure contains information about an error or informational message returned by the server. This structure is defined within the application. CTBDIAG returns a message string and information about the message in this structure.

Client messages are returned to a CLIENTMSG structure, described in the section “CLIENTMSG structure” on page 168.

CLIENTMSG and SERVERMSG structures are part of the Mainframe ClientConnect (MCC) CTPUBLIC copybook.

This structure contains information about all messages received by the client application, including MCC messages, messages returned by the remote transactions, and messages returned by the database (for example, DB2 Access Module messages and Adaptive Server messages).

A SERVERMSG structure is defined as follows:

```
01 SERVER-MSG
05 SMSG-MSGNO          PIC S9(9) COMP SYNC.
05 SMSG-STATE          PIC S9(9) COMP SYNC.
05 SMSG-SEVERITY       PIC S9(9) COMP SYNC.
05 SMSG-TEXT           PIC X(256).
05 SMSG-TEXT-LEN       PIC S9(9) COMP SYNC.
05 SMSG-SVRNAME        PIC X(256).
05 SMSG-SVRNAME-LEN    PIC S9(9) COMP SYNC.
05 SMSG-PROC           PIC X(256).
05 SMSG-PROC-LEN       PIC S9(9) COMP SYNC.
05 SMSG-LINE           PIC S9(9) COMP SYNC.
05 SMSG-STATUS         PIC S9(9) COMP SYNC.
```

- SMSG-MSGNO is the server message number. This field corresponds to the *MESSAGE-NUMBER* argument of the Gateway-Library function TDSNDMSG.
- SMSG-STATE is the message state. This field corresponds to the *ERROR-STATE* argument of the Gateway-Library function TDSNDMSG.
- SMSG-SEVERITY is a symbolic value representing the severity of the message. Severity values are provided in the CTPUBLIC copybook. This field corresponds to the *SEVERITY* argument of the Gateway-Library function TDSNDMSG.

  Table 3-9 lists the legal values for SMSG-SEVERITY.

Table 3-9: Values for the SERVERMSG SMSG-SEVERITY field

SMSG-SEVERITY value	Meaning
CS-SV-INFORM (0)	No error occurred. The message is informational.
CS-SV-API-FAIL (1)	A Client-Library routine generated an error. This error is typically caused by a bad parameter or calling sequence. The server connection is probably salvageable.
CS-SV-RETRY-FAIL (2)	An operation failed, but the operation can be retried.
CS-SV-RESOURCE-FAIL (3)	A resource error occurred. This error is typically caused by an allocation error, a lack of file descriptors, or timeout error. The server connection is probably not salvageable.
CS-SV-CONFIG-FAIL (4)	A configuration error occurred.
CS-SV-COMM-FAIL (5)	An unrecoverable error in the server communication channel occurred. The server connection is not salvageable.
CS-SV-INTERNAL-FAIL (6)	An internal Client-Library error occurred.
CS-SV-FATAL (7)	A serious error occurred. All server connections are unusable.

- SMSG-TEXT is the text of the message string. This field corresponds to the *MESSAGE-TEXT* argument of the Gateway-Library function TDSNDMSG.
- SMSG-TEXT-LEN is the length, in bytes, of SMSG-TEXT. If there is no message text, the value of SMSG-TEXT-LEN is 0. This field corresponds to the *MESSAGE-LENGTH* argument of the Gateway-Library function TDSNDMSG.
- SMSG-SVRNAME is the name of the server that generated the message. This is the server name from the Server Path Table.

  The Server Path Table contains the information needed by Client-Library programs to route requests to a remote server, including the name of the server and connections to use to access that server. This table is part of the Connection Router, described in the Mainframe Connect Client Option *Installation and Administration Guide*.
- SMSG-SVRNAME-LEN is the length, in bytes, of SMSG-SVRNAME.
- SMSG-PROC is the name of the remote procedure or transaction that returned the message—the name of the Adaptive Server stored procedure or the transaction ID of the mainframe transaction. This field corresponds to the *TRANSACTION-ID* argument of the Gateway-Library function TDSNDMSG.
- SMSG-PROC-LEN is the length, in bytes, of SMSG-PROC. This field corresponds to the *TRANSACTION-ID-LENGTH* argument of the Gateway-Library function TDSNDMSG.

- SMSG-LINE is the line number in the called procedure or transaction where the error occurred. It may also be used for miscellaneous information. This field corresponds to the *LINE-ID* argument of the Gateway-Library function TDSNDMSG.
- SMSG-STATUS is reserved for future use.

SQLCA structure

Description

A SQLCA structure can be used in conjunction with CTBDIAG to retrieve Client-Library and server error and informational messages.

```
01 SQLCA-MSG.
05 SQLCAID  PIC X(8).
05 SQLCABC  PIC S9(9) COMP VALUE +0.
05 SQLCODE  PIC S9(9) COMP VALUE +0.
05 SQLERRM.
49 SQLERRML PIC S9(9) COMP VALUE +0.
49 SQLERRMC PIC X(256).
05 SQLERRP  PIC X(8).
05 SQLERRD OCCURS 6 TIMESPIC S9(9).
05 SQLWARN.
10 SQLWARN0 PIC X(1).
10 SQLWARN1 PIC X(1).
10 SQLWARN2 PIC X(1).
10 SQLWARN3 PIC X(1).
10 SQLWARN4 PIC X(1).
10 SQLWARN5 PIC X(1).
10 SQLWARN6 PIC X(1).
10 SQLWARN7 PIC X(1).
05 SQLEXT   PIC X(8).
```

- *SQLCAID* is “SQLCA” (This value is automatically provided).
- *SQLCABC* is ignored.
- *SQLCODE* is the server or Client-Library message number.
  For information on how Client-Library maps message numbers to SQLCODE, see “SQLCODE structure” on page 198. For a list of gateway messages, see the *Messages and Codes* for Open ServerConnect and Open ClientConnect shipped with this product.
- *SQLERRML* is the length of the actual message text (not the length of the text placed in *SQLERRMC*).
- *SQLERRMC* is the null-terminated text of the message. If the message is too long for the array, Client-Library truncates it before appending the null terminator.
- *SQLERRP* is the first eight characters of the stored procedure, if any, being executed at the time of the error.
- *SQLERRD* is the number of rows successfully inserted, updated, or deleted before the error occurred.
- *SQLEXT* is ignored.
- *SQLWARN* is an array of warnings:

- If *SQLWARN0* is blank, all other *SQLWARN* variables are blank. If *SQLWARN0* is not blank, at least one other SQLWARN variable is set to W.
- If *SQLWARN1* is W, Client-Library truncated at least one column's value when storing it into a mainframe variable.
- If *SQLWARN2* is W, at least one null value was eliminated from the argument set of a function.
- If *SQLWARN3* is W, the number of mainframe variables specified in the into clause of a select statement is not equal to the number of result columns.
- If *SQLWARN4* is W, a dynamic SQL update or delete statement did not include a where clause.
- If *SQLWARN5* is W, a server conversion or truncation error occurred.

SQLCODE structure

Description

A SQLCODE structure can be used in conjunction with CTBDIAG to retrieve Client-Library and server error and informational messages. A SQLCODE structure can be located anywhere and mapped to SQLCA.

A SQLCODE structure is defined as a 4-byte integer.

Client-Library always sets SQLCODE and the SQLCODE field of the SQLCA structure identically (See "SQLCA structure" on page 197).

Mapping server messages to SQLCODE

A server message number is mapped to a SQLCODE of 0 if it has a severity of 0.

Other server messages can be mapped to a SQLCODE of 0 as well.

Server message numbers are negated before being placed into SQLCODE. This ensures that SQLCODE is negative if an error occurs.

For a list of server messages returned by gateway products (Mainframe ClientConnect, Open ServerConnect, or OmniSQL Access Module for DB2), see the Mainframe Connect Client Option and Server Option *Messages and Codes*.

Mapping Client-Library messages to SQLCODE

The Client-Library message "No rows affected" is mapped to a SQLCODE of 100.

Client-Library messages with CS-SV-INFORM severities are mapped to a SQLCODE of 0.

Other Client-Library messages may be mapped to a SQLCODE of 0 as well.

Client-Library message numbers are negated before being placed into SQLCODE. This ensures that SQLCODE is negative when an error occurs.

For a list of Client-Library messages, see the Mainframe Connect Client Option and Server Option *Messages and Codes*.

Handles

Client-Library uses handles at three levels. Each handle defines and manages a particular environment. Each type of handle can have certain properties, listed below.

Note Most Client-Library functions include a handle argument. An application must allocate these handles before using them as arguments.

Types of handles

The following handles are used with Client-Library:

- *Context handle*. A context handle defines a particular application, context, or operating environment. The context handle is defined in the program call CSBCTXALLOC.

  An application can have only one context.

  A context handle corresponds to the *IHANDLE* structure in the Open ServerConnect Gateway-Library.

  The context handle can have the following properties listed in Table 3-10 on page 199:

Table 3-10: Context properties

Property	Set by
CS-EXTRA-INF	CSBCONFIG
CS-LOGIN-TIMEOUT	CTBCONFIG

Property	Set by
CS-MAX-CONNECT	CTBCONFIG
CS-NETIO	CTBCONFIG
CS-NOINTERRUPT	CTBCONFIG
CS-TEXTLIMIT	CTBCONFIG
CS-TIMEOUT	CTBCONFIG
CS-VERSION	CSBCONFIG

- *Connection handle*. This is the handle for an individual client/server connection. The connection handle is defined in the program call CTBCONALLOC. If parallel sessions are used, there must be one connection handle for each session. An application can have up to 25 connections.

  Open ClientConnect uses a Connection Router program to define connections. Each connection handle corresponds to a connection defined with the Connection Router. For details about the Connection Router, see the Mainframe Connect Client Option *Installation and Administration Guide*.

  A connection handle can have the following properties listed in Table 3-11.

Table 3-11: Connection properties

Property	Set by
CS-APPNAME	CTBCONPROPS
CS-CHARSETCNV	CTBCONPROPS
CS-COMMBLOCK	CTBCONPROPS
CS-EXTRA-INF	CTBCONPROPS
CS-HOSTNAME	CTBCONPROPS
CS-LOC-PROP	CTBCONPROPS
CS-LOGIN-STATUS	CTBCONPROPS
CS-NET-DRIVER	CTBCONPROPS
CS-NETIO	CTBCONPROPS
CS-NOINTERRUPT	CTBCONPROPS
CS-PACKETSIZE	CTBCONPROPS
CS-PASSWORD	CTBCONPROPS
CS-TDS-VERSION	CTBCONPROPS
CS-TEXTLIMIT	CTBCONPROPS
CS-TRANSACTION-NAME	CTBCONPROPS
CS-USERNAME	CTBCONPROPS

- *Command handle*. A command handle defines a command space, which is used to send commands to a server over a connection and process the results. A command handle is defined in the program's CTBCMDALLOC call. Each command handle is associated with a particular connection. There can be any number of command handles associated with a connection.

  A command handle and its associated connection handle correspond to the *TDPROC* handle in Open ServerConnect's Gateway-Library.

  A command handle can have the following property:

  - CS-USERDATA

Routines that affect handles

Table 3-12 lists the routines that allocate, use, and deallocate handles.

Table 3-12: Routines that manipulate hidden structures

Structure	Allocated and used by
CONTEXT	CSBCTXALLOC, CTBCONFIG, CSBCONFIG, CSBCTXDROP
CONNECTION	CTBCONALLOC, CTBCONPROPS, CTBCONDROP
COMMAND	CTBCMDALLOC, CTBCMDPROPS, CTBCMDDROP

APPENDIX A

Sample Language Requests

This appendix contains three Open ClientConnect application program samples that send a language request to an Adaptive Server Enterprise. They retrieve information from tables on the target server.

These sample programs are provided as part of the Open ClientConnect package, on the API tape. The Transaction Router Service (TRS) administrator can create the tables SYBASE.SAMPLETB and SYBASE.NEWTABLE on that server with scripts provided with TRS.

The following sample programs are included in this appendix:

- Sample program - SYCTSAA5

  Demonstrates how to send a language request to an Adaptive Server Enterprise.

- SYCTSAP5 - sample language request

  Demonstrates explicit conversions of different datatypes.

- SYCTSAT5 - sample language request

  Demonstrates implicit conversions of different datatypes.

Sample program - SYCTSAA5

```
*@(#) syctsaa5.cobol 11.2 12/14/95        */
 ****************************************************************
 *
 * Confidential property of Sybase, Inc.
 * (c) Copyright Sybase, Inc. 1985 TO 1997.
 * All rights reserved.
 *
 ****************************************************************
 ******* SYCTSAA5 - Client Language Request APPL - COBOL - CICS **
 **
 **  CICS TRANID: SYA5
```

```
**
**  PROGRAM:     SYCTSAA5
**
**  PURPOSE:  Demonstrates Open Client for CICS CALLs.
**
**  FUNCTION: Illustrates how to send a language request with
**            parameters to:
**
**              - A SQL Server
**
**            SQL Server:
**
**              If the request is sent to a SQL Server it
**              executes the SQL statement:
**
**                  SELECT  FIRSTNME, EDUCLVL
**                    FROM  SYBASE.SAMPLETB
**
**  PREREQS:  Before running SYCTSAA5, make sure that the server
**            you wish to access has an entry in the Connection
**            Router Table for that Server and the MCG(s) that
**            you wish to use.
**
**  INPUT:    On the input screen, make sure to enter the Server
**            name, user id, and password for the target server.
**            TRAN NAME is not used for LAN servers.
**
**
**  Open Client CALLs used in this sample:
**
**    CSBCONVERT    convert a datatype from one value to another
**    CSBCTXALLOC   allocate a context
**    CSBCTXDROP    drop a context
**    CTBBIND       bind a column variable
**    CTBCLOSE      close a server connection
**    CTBCONFIG     set or retrieve context properties
**    CTBCMDALLOC   allocate a command
**    CTBCMDDROP    drop a command
**    CTBCOMMAND    initiate remote procedure CALL
**    CTBCONALLOC   allocate a connection
**    CTBCONDROP    drop a connection
**    CTBCONPROPS   alter properties of a connection
**    CTBCONNECT    open a server connection
**    CTBDESCRIBE   return a description of RESULT data
**    CTBDIAG       retrieve SQLCODE messages
**    CTBEXIT       exit client library
```

```
**    CTBFETCH      FETCH RESULT data
**    CTBINIT       init client library
**    CTBPARAM      define a command PARAMETER
**    CTBRESULTS    set up RESULT data
**    CTBRESINFO    return RESULT set info
**    CTBSEND       send a request TO the server
**
** History:
**
** Date    BTS#   Descrition
** ======= ====== ============================================
** Feb1795        Create
** Oct1895 99999  Rewrite and add front end to the program
**
**
*****************************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  SYCTSAA5.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER.  xyz.
 OBJECT-COMPUTER.  xyz.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*****************************************************************
** Client Library Cobol Copy Book
*****************************************************************
   COPY CTPUBLIC.
*****************************************************************
** CICS BMS DEFINITIONS
*****************************************************************
    COPY SYCTBA5.

*****************************************************************
 *  Standard CICS Attribute and Print Control Chararcter List
*****************************************************************
    COPY DFHBMSCA.
*****************************************************************
** CICS Standard Attention Identifiers Cobol Copy Book
*****************************************************************
    COPY DFHAID.
*****************************************************************
*    CONSTANTS
*****************************************************************
 01  C-N                         PIC X(01) VALUE 'N'.
 01  C-Y                         PIC X(01) VALUE 'Y'.
```

```
 01  I1                          PIC S9(9) COMP SYNC VALUE IS 0.
 01  MAX-SCREEN-ROWS             PIC S9(4) VALUE +10.
 01  MSG-TEXT-1                  PIC X(70) VALUE ' '.
 01  MSG-TEXT-2                  PIC X(70)
                                 VALUE 'Press Clear To Exit'.
*****************************************************************
*    OPEN CLIENT VARIABLES
*****************************************************************
 01  OUTLEN                      PIC S9(9) COMP VALUE +0.
 01  RESTYPE                     PIC S9(9) COMP VALUE +0.
 01  NETDRIVER                   PIC S9(9) COMP VALUE +9999.
 01  PAGE-CNT                    PIC S9(4) COMP VALUE +0.
 01  UTIME                       PIC S9(15) COMP-3.
 01  TMP-DATE                    PIC X(08).
 01  TMP-TIME                    PIC X(08).
 01  ENTER-DATA-SW               PIC X(01) VALUE 'N'.
**---------------------------------------------------------------
** WORK AREAS
**---------------------------------------------------------------
  01  NO-MORE-MSGS-SW             PIC X(01).
     88  NO-MORE-MSGS VALUE 'Y'.
 01  NO-ERRORS-SW                PIC X(01).
     88  NO-ERRORS    VALUE 'N'.
 01  SWITCHES.
     05  SW-RESULTS                PIC X(01) VALUE 'Y'.
         88  NO-MORE-RESULTS VALUE 'N'.
     05  SW-FETCH                  PIC X(01) VALUE 'Y'.
         88  NO-MORE-ROWS VALUE 'N'.
     05  SW-DIAG                   PIC X(01) VALUE 'N'.
         88  DIAG-MSGS-INITIALIZED VALUE 'Y'.
 01  INTERNAL-FIELDS.
     05  I                     PIC S9(9) COMP.
     05  CF-FOUR               PIC S9(9) COMP VALUE +4.
     05  CF-LANG2-SIZE         PIC S9(9) COMP VALUE +45.
     05  DATA-SMALLINT         PIC S9(4) COMP VALUE +4.
 01  CS-LIB-MISC-FIELDS.
     05  CSL-CMD-HANDLE        PIC S9(9) COMP VALUE +0.
     05  CSL-CON-HANDLE        PIC S9(9) COMP VALUE +0.
     05  CSL-CTX-HANDLE        PIC S9(9) COMP VALUE +0.
     05  CSL-RC                PIC S9(9) COMP VALUE +0.
  01  PROPS-FIELDS.
     05  PF-SERVER             PIC X(30) VALUE IS SPACES.
     05  PF-SERVER-SIZE        PIC S9(9) COMP VALUE +0.
     05  PF-USER               PIC X(08) VALUE IS SPACES.
     05  PF-USER-SIZE          PIC S9(9) COMP VALUE +0.
     05  PF-PWD                PIC X(08) VALUE IS SPACES.
```

```
    05  PF-PWD-SIZE           PIC S9(9) COMP VALUE +0.
    05  PF-TRAN               PIC X(08) VALUE IS SPACES.
    05  PF-TRAN-SIZE          PIC S9(9) COMP VALUE +0.
    05  PF-NETDRV             PIC X(08) VALUE IS SPACES.
    05  PF-DRV-SIZE           PIC S9(9) COMP VALUE +0.
    05  PF-STRLEN             PIC S9(9) COMP.
    05  PF-MSGLIMIT           PIC S9(9) COMP.
01  DIAG-FIELDS.
    05  DG-MSGNO              PIC S9(9) COMP VALUE +1.
    05  DG-NUM-OF-MSGS        PIC S9(9) COMP VALUE +0.
01  CONFIG-FIELDS.
    05  CF-MAXCONNECT         PIC S9(9) COMP.
    05  CF-OUTLEN             PIC S9(9) COMP.

01  FETCH-FIELDS.
    05  FF-ROWS-READ          PIC S9(9) COMP.
    05  FF-ROW-NUM            PIC S9(9) COMP VALUE +0.

01  RESINFO-FIELDS.
    05  RF-NUMDATA            PIC S9(9) COMP.
    05  RF-NUMDATA-SIZE       PIC S9(9) COMP VALUE +4.

01  OUTPUT-ROW.
    05  OR-COL-FIRSTNME-CHAR  PIC X(12).
    05  SPACE1                PIC X(01) VALUE ' '.
    05  OR-COL-EDUCLVL        PIC 9(3).

01  OUTPUT-ROW-STR REDEFINES OUTPUT-ROW PIC X(16).

01  OUTPUT-ROW-SIZE           PIC S9(4) COMP VALUE +16.

01  OUTPUT-ROW2.
    05  OR2-MESG              PIC X(37)
                  VALUE 'The maximum number of connections is '.
    05  OR2-MAXCONNECT        PIC ZZZZ9.
    05  OR2-PERIOD            PIC X(01) VALUE '.'.

01  OUTPUT-ROW-STR2 REDEFINES OUTPUT-ROW2 PIC X(43).

01  OUTPUT-ROW2-SIZE          PIC S9(4) COMP VALUE +43.

01  OUTPUT-ROW4.
    05  OR4-MESG              PIC X(25)
                              VALUE 'The number of columns is '.
    05  OR4-NUMDATA           PIC ZZZZ9.
    05  OR4-PERIOD            PIC X(01)     VALUE '.'.
```

```
01  OUTPUT-ROW-STR4 REDEFINES OUTPUT-ROW4 PIC X(31).

01  OUTPUT-ROW4-SIZE          PIC S9(4) COMP VALUE +31.

01  COLUMN-FIELDS.
    05  CF-COL-FIRSTNME.
        10 CF-COL-FIRSTNME-LL PIC S9(4) COMP.
        10 CF-COL-FIRSTNME-TXT PIC X(12).
    05  CF-COL-FIRSTNME-CHAR  PIC X(12).
    05  CF-COL-EDUCLVL        PIC S9(4) COMP.
    05  CF-COL-LEN            PIC S9(9) COMP.
    05  CF-COL-NULL           PIC S9(9) COMP VALUE +0.
    05  CF-COL-NUMBER         PIC S9(9) COMP VALUE +1.
    05  CF-COL-INDICATOR      PIC S9(4) COMP VALUE +0.

01  LANG-FIELDS.
    05  CF-LANG1              PIC X(20)
        VALUE 'Wrong SQL statement'.
    05  CF-LANG2              PIC X(45)
        VALUE 'SELECT FIRSTNME, EDUCLVL FROM SYBASE.SAMPLETB'.
    05  filler                PIC X(01) VALUE LOW-VALUE.

01  MSG-FIELDS.
    05  MF-CANCELED           PIC X(16)
                              VALUE 'Cancel requested'.
    05  MF-CANCELED-SIZE      PIC S9(9) COMP VALUE +16.

01  DATAFMT.
    05  DF-NAME               PIC X(132).
    05  DF-NAMELEN            PIC S9(9) COMP.
    05  DF-DATATYPE           PIC S9(9) COMP.
    05  DF-FORMAT             PIC S9(9) COMP.
    05  DF-MAXLENGTH          PIC S9(9) COMP.
    05  DF-SCALE              PIC S9(9) COMP.
    05  DF-PRECISION          PIC S9(9) COMP.
    05  DF-STATUS             PIC S9(9) COMP.
    05  DF-COUNT              PIC S9(9) COMP.
    05  DF-USERTYPE           PIC S9(9) COMP.
    05  DF-LOCALE             PIC X(68).

01  DATAFMT2.
    05  DF2-NAME              PIC X(132).
    05  DF2-NAMELEN           PIC S9(9) COMP.
    05  DF2-DATATYPE          PIC S9(9) COMP.
    05  DF2-FORMAT            PIC S9(9) COMP.
```

```
     05  DF2-MAXLENGTH          PIC S9(9) COMP.
     05  DF2-SCALE              PIC S9(9) COMP.
     05  DF2-PRECISION          PIC S9(9) COMP.
     05  DF2-STATUS             PIC S9(9) COMP.
     05  DF2-COUNT              PIC S9(9) COMP.
     05  DF2-USERTYPE           PIC S9(9) COMP.
     05  DF2-LOCALE             PIC X(68).

 01 DISP-MSG.
    05 TEST-CASE               PIC X(08) VALUE IS 'SYCTSAA5'.
    05 FILLER                  PIC X(01) VALUE IS SPACES.
    05 MSG.
       10 SAMP-LIT             PIC X(05) VALUE IS 'rc = '.
       10 SAMP-RC              PIC -Z9.
       10 FILLER               PIC X(02) VALUE IS ', '.
       10 REST-LIT             PIC X(12) VALUE IS
                                  'Result Type:'.
       10 REST-TYPE            PIC Z(3)9.
       10 FILLER               PIC X(03) VALUE IS SPACES.
       10 MSGSTR               PIC X(40) VALUE IS SPACES.

*******************************
** Client Message Structure  **
*******************************

 01  CLIENT-MSG.
     05  CM-SEVERITY           PIC S9(9) COMP SYNC.
     05  CM-MSGNO              PIC S9(9) COMP SYNC.
     05  CM-TEXT               PIC X(256).
     05  CM-TEXT-LEN           PIC S9(9) COMP SYNC.
     05  CM-OS-MSGNO           PIC S9(9) COMP SYNC.
     05  CM-OS-MSGTXT          PIC X(256).
     05  CM-OS-MSGTEXT-LEN     PIC S9(9) COMP SYNC.
     05  CM-STATUS             PIC S9(9) COMP.

 01  DISP-CLIENT-MSG-HDR.
     05  CLIENT-MSG-HDR        PIC X(15) VALUE IS
                                   'Client Message:'.

 01  DISP-CLIENT-MSG-1.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  CM-SEVERITY-HDR       PIC X(09) VALUE IS 'Severity:'.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  CM-SEVERITY-DATA      PIC Z(8)9.
     05  CM-STATUS-HDR         PIC X(12) VALUE IS
                                   ',  Status:  '.
```

```
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  CM-STATUS-DATA         PIC Z(8)9.

01  DISP-CLIENT-MSG-2.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  CM-OC-MSGNO-HDR        PIC X(09) VALUE IS 'OC MsgNo:'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  CM-OC-MSGNO-DATA       PIC Z(8)9.

01  DISP-CLIENT-MSG-3.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  CM-OC-MSG-HDR          PIC X(09) VALUE IS 'OC MsgTx:'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  CM-OC-MSG-DATA         PIC X(66).

01  DISP-CLIENT-MSG-3A.
    05  CM-OC-MSG-DATA-1       PIC X(66).
    05  CM-OC-MSG-DATA-2       PIC X(66).
    05  CM-OC-MSG-DATA-3       PIC X(66).
    05  CM-OC-MSG-DATA-4       PIC X(58).

01  DISP-CLIENT-MSG-3B.
    05  FILLER                 PIC X(13) VALUE IS SPACES.
    05  CM-OC-MSG-DATA-X       PIC X(66).

01  DISP-EMPTY-CLIENT-MSG-3.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  CM-OC-MSG-HDR          PIC X(09) VALUE IS 'OC MsgTx:'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  NO-DATA                PIC X(11) VALUE IS 'No Message!'.

01  DISP-CLIENT-MSG-4.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  CM-OS-MSG-HDR          PIC X(09) VALUE IS 'OS MsgNo:'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  CM-OS-MSGNO-DATA       PIC Z(8)9.

01  DISP-CLIENT-MSG-5.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  CM-OS-MSG-HDR          PIC X(09) VALUE IS 'OS MsgTx:'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  CM-OS-MSG-DATA         PIC X(66).

01  DISP-CLIENT-MSG-5A.
    05  CM-OS-MSG-DATA-1       PIC X(66).
    05  CM-OS-MSG-DATA-2       PIC X(66).
```

```
     05  CM-OS-MSG-DATA-3      PIC X(66).
     05  CM-OS-MSG-DATA-4      PIC X(58).

 01  DISP-EMPTY-CLIENT-MSG-5.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  CM-OS-MSG-HDR         PIC X(09) VALUE IS 'OS MsgTx:'.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  NO-DATA               PIC X(11) VALUE IS 'No Message!'.

*******************************
** Server Message Structure  **
*******************************

 01  SERVER-MSG.
     05  SM-MSGNO              PIC S9(9) COMP.
     05  SM-STATE              PIC S9(9) COMP.
     05  SM-SEV                PIC S9(9) COMP.
     05  SM-TEXT               PIC X(256).
     05  SM-TEXT-LEN           PIC S9(9) COMP.
     05  SM-SVRNAME            PIC X(256).
     05  SM-SVRNAME-LEN        PIC S9(9) COMP.
     05  SM-PROC               PIC X(256).
     05  SM-PROC-LEN           PIC S9(9) COMP.
     05  SM-LINE               PIC S9(9) COMP.
     05  SM-STATUS             PIC S9(9) COMP.

 01  DISP-SERVER-MSG-HDR.
     05  SERVER-MSG-HDR        PIC X(15) VALUE IS
                                    'Server Message:'.

 01  DISP-SERVER-MSG-1.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  SM-MSG-NO-HDR         PIC X(09) VALUE IS
                                    'Message#:'.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  SM-MSG-NO-DATA        PIC Z(8)9.
     05  SM-SEVERITY-HDR       PIC X(12) VALUE IS
                                    ',  Severity:'.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  SM-SEVERITY-DATA      PIC Z(8)9.
     05  SM-STATE-HDR          PIC X(12) VALUE IS
                                    ',  State No:'.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  SM-STATE-DATA         PIC Z(8)9.

 01  DISP-SERVER-MSG-2.
```

```
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-LINE-NO-HDR         PIC X(09) VALUE IS
                                    'Line  No:'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-LINE-NO-DATA        PIC Z(8)9.
    05  SM-STATUS-HDR          PIC X(12) VALUE IS
                                    ',  Status  :'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-STATUS-DATA         PIC Z(8)9.

01  DISP-SERVER-MSG-3.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-SVRNAME-HDR         PIC X(09) VALUE IS 'Serv Nam:'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-SVRNAME-DATA        PIC X(66).
    05  FILLER                 PIC X(03) VALUE IS '...'.

01  DISP-SERVER-MSG-4.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-PROC-ID-HDR         PIC X(09) VALUE IS 'Proc  ID:'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-PROC-ID-DATA        PIC X(66).

01  DISP-SERVER-MSG-5.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-MSG-HDR             PIC X(09) VALUE IS 'Message :'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-MSG-DATA            PIC X(66).

01  DISP-SERVER-MSG-5A.
    05  SM-MSG-DATA-1          PIC X(66).
    05  SM-MSG-DATA-2          PIC X(66).
    05  SM-MSG-DATA-3          PIC X(66).
    05  SM-MSG-DATA-4          PIC X(58).

01  DISP-SERVER-MSG-5X.
    05  FILLER                 PIC X(13) VALUE IS SPACES.
    05  SM-MSG-DATA-X          PIC X(66).

01  CICS-FIELDS.
    05  CICS-RESPONSE        PIC S9(9) COMP.

01  QUERY-FIELDS.
    05  QF-LEN                 PIC S9(4) COMP VALUE +1.
    05  QF-MAXLEN              PIC S9(4) COMP VALUE +1.
    05  QF-ANSWER              PIC X(01) VALUE IS SPACES.
```

```
 PROCEDURE DIVISION.
**********************

**************************
* CICS Condition Handler *
**************************

     EXEC CICS HANDLE CONDITION MAPFAIL(NO-INPUT)
                                ERROR(ERRORS)
     END-EXEC.

********************
* CICS Aid Handler *
********************

     EXEC CICS HANDLE AID ANYKEY(NO-INPUT)
                          CLEAR(GETOUT)
     END-EXEC.

*****************************
*    PROGRAM INITIALIZATION *
*****************************

     MOVE C-N    TO NO-MORE-MSGS-SW.
     MOVE C-N    TO NO-ERRORS-SW.
     MOVE C-Y    TO SW-DIAG.

     COMPUTE PAGE-CNT = PAGE-CNT + 1.

     PERFORM GET-SYSTEM-TIME.

     MOVE LOW-VALUES TO A5PANELO.
     MOVE -1         TO SERVERL.

 GET-INPUT-AGAIN.


     PERFORM DISPLAY-INITIAL-SCREEN.

     PERFORM GET-INPUT-DATA.

**********************************
*    ALLOCATE A CONTEXT STRUCTURE *
**********************************
```

```
      MOVE ZERO TO CSL-CTX-HANDLE.

      CALL 'CSBCTXAL' USING CS-VERSION-50
                            CSL-RC
                            CSL-CTX-HANDLE.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
           MOVE SPACES TO MSGSTR
           STRING 'CSBCTXAL failed' DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
      END-IF.

*********************************
* INTITIALIZE THE CLIENT-LIBRARY *
*********************************

      CALL 'CTBINIT' USING CSL-CTX-HANDLE
                           CSL-RC
                           CS-VERSION-50.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBINIT failed' DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
      END-IF.

      PERFORM PROCESS-INPUT.

      PERFORM QUIT-CLIENT-LIBRARY.

      GOBACK.

*============================================================
*==                                                        ==
*== Subroutine to get system date/time                     ==
*==                                                        ==
*============================================================
 GET-SYSTEM-TIME.
*----------------

      EXEC CICS ASKTIME
                ABSTIME(UTIME)
```

```
     END-EXEC.

     EXEC CICS FORMATTIME
               ABSTIME(UTIME)
               DATESEP('/')
               MMDDYY(TMP-DATE)
               TIME(TMP-TIME)
               TIMESEP
     END-EXEC.

*=========================================================
*==                                                     ==
*== Subroutine to display SYA5 initial screen           ==
*==                                                     ==
*=========================================================
 DISPLAY-INITIAL-SCREEN.
*----------------------

     MOVE TMP-DATE   TO SDATEO.
     MOVE TMP-TIME   TO STIMEO.
     MOVE 'SYCTSAA5' TO PROGNMO.

     MOVE PAGE-CNT   TO SPAGEO.
     MOVE MSG-TEXT-1 TO MSG1O.
     MOVE MSG-TEXT-2 TO MSG2O.

     EXEC CICS SEND MAP('A5PANEL')
                    MAPSET('SYCTBA5')
                    CURSOR
                    FRSET
                    ERASE
                    FREEKB
     END-EXEC.

*=========================================================
*==                                                     ==
*== Subroutine to get input data                        ==
*==                                                     ==
*=========================================================
 GET-INPUT-DATA.
*---------------

     EXEC CICS RECEIVE MAP('A5PANEL')
                    MAPSET('SYCTBA5')
                    ASIS
     END-EXEC.
```

```
IF SERVERL = ZERO
  THEN
    IF PF-SERVER = SPACES
      THEN
        MOVE 'Please Enter Server Name' TO MSG-TEXT-1
        MOVE -1                         TO SERVERL
        MOVE C-Y                        TO ENTER-DATA-SW
    END-IF
  ELSE
    MOVE SERVERI   TO PF-SERVER
    MOVE SERVERL   TO PF-SERVER-SIZE
END-IF.

IF USERL = ZERO
  THEN
    IF PF-USER = SPACES
      THEN
        MOVE 'Please Enter User-ID' TO MSG-TEXT-1
        MOVE -1                     TO USERL
        MOVE C-Y                    TO ENTER-DATA-SW
    END-IF
  ELSE
    MOVE USERI   TO PF-USER
    MOVE USERL   TO PF-USER-SIZE
    MOVE PF-USER TO USERO
END-IF.

IF PSWDL NOT EQUAL ZERO
  THEN
    MOVE PSWDI TO PF-PWD
    MOVE PSWDL TO PF-PWD-SIZE
END-IF.

IF TRANL NOT EQUAL ZERO
  THEN
    MOVE TRANI TO PF-TRAN
    MOVE TRANL TO PF-TRAN-SIZE
END-IF.

IF NETDRVL NOT EQUAL ZERO
  THEN
    MOVE NETDRVI TO PF-NETDRV
    MOVE NETDRVL TO PF-DRV-SIZE
END-IF.
```

```
           IF ENTER-DATA-SW = C-Y
             THEN
               MOVE C-N TO ENTER-DATA-SW
               PERFORM DISPLAY-INITIAL-SCREEN
               PERFORM GET-INPUT-DATA
           END-IF.

       *============================================================
       *==                                                        ==
       *== Subroutine to process input data                       ==
       *==                                                        ==
       *============================================================
        PROCESS-INPUT.


       *********************************
       * ALLOCATE A CONNECTION HANDLE. *
       *********************************

           MOVE ZERO TO CSL-CON-HANDLE.

           CALL 'CTBCONAL' USING CSL-CTX-HANDLE
                                 CSL-RC
                                 CSL-CON-HANDLE.

           IF CSL-RC NOT EQUAL CS-SUCCEED
             THEN
                MOVE SPACES TO MSGSTR
                STRING 'CTBCONAL failed' DELIMITED BY SIZE INTO MSGSTR
                PERFORM PRINT-MSG
                PERFORM ALL-DONE
           END-IF.

       ******************
       * SET THE USER ID *
       ******************
CALL 'CTBCONPR' USING CSL-CON-HANDLE
                                 CSL-RC
                                 CS-SET
                                 CS-USERNAME
                                 PF-USER
                                 PF-USER-SIZE
                                 CS-FALSE
                                 OUTLEN.
           IF CSL-RC NOT EQUAL CS-SUCCEED
             THEN
```

```
           MOVE SPACES TO MSGSTR
           STRING 'CTBCONPR for user-id failed' DELIMITED BY SIZE
                                                INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
      END-IF.
********************
* SET THE PASSWORD *
********************
      CALL 'CTBCONPR' USING CSL-CON-HANDLE
                            CSL-RC
                            CS-SET
                            CS-PASSWORD
                            PF-PWD
                            PF-PWD-SIZE
                            CS-FALSE
                            OUTLEN.
      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBCONPR for password failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
      END-IF.

********************
* SET THE TRAN NAME *
********************

      IF PF-TRAN-SIZE IS NOT EQUAL TO ZEROES THEN

          CALL 'CTBCONPR' USING CSL-CON-HANDLE
                                CSL-RC
                                CS-SET
                                CS-TRANSACTION-NAME
                                PF-TRAN
                                PF-TRAN-SIZE
                                CS-FALSE
                                OUTLEN

          IF CSL-RC NOT EQUAL CS-SUCCEED
            THEN
               MOVE SPACES TO MSGSTR
               STRING 'CTBCONPR for TRAN name failed'
                      DELIMITED BY SIZE INTO MSGSTR
```

```
               PERFORM PRINT-MSG
               PERFORM ALL-DONE
             END-IF

     END-IF.

*****************************
* SET THE NET DRIVER PROPERTY *
*****************************

     IF PF-NETDRV = SPACES OR PF-NETDRV = 'LU62'                X
                          OR PF-NETDRV = 'lu62'
         MOVE CS-LU62 TO NETDRIVER
     ELSE
       IF PF-NETDRV = 'IBMTCPIP' OR PF-NETDRV = 'ibmtcpip'
         MOVE CS-TCPIP TO NETDRIVER
     ELSE
       IF PF-NETDRV = 'INTERLIN' OR PF-NETDRV = 'interlin'
         MOVE CS-INTERLINK TO NETDRIVER
     ELSE
       IF PF-NETDRV = 'CPIC' OR PF-NETDRV = 'cpic'
         MOVE CS-NCPIC TO NETDRIVER
     END-IF.

     IF PF-DRV-SIZE IS NOT EQUAL TO ZEROES THEN

         CALL 'CTBCONPR' USING CSL-CON-HANDLE
                               CSL-RC
                               CS-SET
                               CS-NET-DRIVER
                               NETDRIVER
                               CS-UNUSED
                               CS-FALSE
                               OUTLEN

         IF CSL-RC NOT EQUAL CS-SUCCEED
           THEN
              MOVE SPACES TO MSGSTR
              STRING 'CTBCONPR for network driver failed'
                     DELIMITED BY SIZE INTO MSGSTR
              PERFORM PRINT-MSG
              PERFORM ALL-DONE
           END-IF

     END-IF.
```

```
      *********************************
      * SET FOR MAINFRAME EXTRA INFO *
      *********************************

           CALL 'CTBCONPR' USING CSL-CON-HANDLE
                                 CSL-RC
                                 CS-SET
                                 CS-EXTRA-INF
                                 CS-TRUE
                                 CS-UNUSED
                                 CS-FALSE
                                 CS-UNUSED.

           IF CSL-RC NOT EQUAL CS-SUCCEED
             THEN
                MOVE SPACES TO MSGSTR
                STRING 'CTBCONPR for extra info failed'
                                   DELIMITED BY SIZE INTO MSGSTR
                PERFORM PRINT-MSG
                PERFORM ALL-DONE
           END-IF.

      **********************************
      * SETUP retrieval of All Messages *
      **********************************

           CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                                CSL-RC,
                                CS-UNUSED,
                                CS-INIT,
                                CS-ALLMSG-TYPE,
                                CS-UNUSED,
                                CS-UNUSED.

           IF CSL-RC NOT EQUAL CS-SUCCEED
             THEN
               MOVE SPACES TO MSGSTR
               STRING 'CTBDIAG CS-INIT failed' DELIMITED BY SIZE
                                                    INTO MSGSTR
               PERFORM PRINT-MSG
               PERFORM ALL-DONE
           END-IF.



      *****************************************
```

```
* set the upper limit of number of messages *
*********************************************

     MOVE 5 TO PF-MSGLIMIT.

     CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                          CSL-RC,
                          CS-UNUSED,
                          CS-MSGLIMIT,
                          CS-ALLMSG-TYPE,
                          CS-UNUSED,
                          PF-MSGLIMIT.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBDIAG CS-MSGLIMIT failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

************************
* CONNECT TO THE SERVER *
************************

     CALL 'CTBCONNE' USING CSL-CON-HANDLE
                           CSL-RC
                           PF-SERVER
                           PF-SERVER-SIZE
                           CS-FALSE.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBCONNE failed' DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
     END-IF.

     IF NO-ERRORS
       THEN
         PERFORM SEND-COMMAND
     END-IF.

***********************************
```

```
* PROCESS THE RESULTS OF THE COMMAND *
*************************************

     IF NO-ERRORS
       THEN
         PERFORM RESULTS-PROCESSING UNTIL NO-MORE-RESULTS
         PERFORM CLOSE-CONNECTION
     END-IF.

 PROCESS-INPUT-EXIT.
     EXIT.

*============================================================
*==                                                        ==
*== Subroutine to allocate, send, and process commands     ==
*==                                                        ==
*============================================================
 SEND-COMMAND.

*-------------------------------------------------------------
*  find out what the maximum number of connections is
*-------------------------------------------------------------
     CALL 'CTBCONFI'  USING CSL-CTX-HANDLE,
                            CSL-RC,
                            CS-GET,
                            CS-MAX-CONNECT,
                            CF-MAXCONNECT,
                            CF-FOUR,
                            CS-FALSE,
                            CF-OUTLEN.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBCONFI CS-GET failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

*-------------------------------------------------------------
*  allocate a command handle
*-------------------------------------------------------------

     CALL 'CTBCMDAL' USING CSL-CON-HANDLE,
                           CSL-RC,
```

```
                              CSL-CMD-HANDLE.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBCMDAL failed' DELIMITED BY SIZE
                                                  INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.



     *-------------------------------------------------------------
      *    prepare the language request
      *-------------------------------------------------------------

      MOVE CF-LANG2-SIZE TO PF-STRLEN.

      CALL 'CTBCOMMA' USING CSL-CMD-HANDLE,
                            CSL-RC,
                            CS-LANG-CMD,
                            CF-LANG2,
                            PF-STRLEN,
                            CS-UNUSED.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBCOMMA CS-LANG-CMD failed' DELIMITED BY SIZE
                                                  INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.

      *-------------------------------------------------------------
      *    send the language request
      *-------------------------------------------------------------
      CALL 'CTBSEND' USING CSL-CMD-HANDLE,
                           CSL-RC.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBSEND failed' DELIMITED BY SIZE
                                                  INTO MSGSTR
```

```
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
      END-IF.

 SEND-COMMAND-EXIT.
      EXIT.

*=========================================================
*==                                                     ==
*== Subroutine to process result                        ==
*==                                                     ==
*=========================================================
 RESULTS-PROCESSING.

***************************
* SET UP THE RESULTS DATA *
***************************

      CALL 'CTBRESUL' USING CSL-CMD-HANDLE
                            CSL-RC
                            RESTYPE.

************************************************
* DETERMINE THE OUTCOME OF THE COMMAND EXECUTION *
************************************************

      EVALUATE CSL-RC

          WHEN CS-SUCCEED

*****************************************************************
* DETERMINE THE TYPE OF RESULT RETURNED BY THE CURRENT REQUEST *
*****************************************************************

              EVALUATE RESTYPE

***********************
* PROCESS ROW RESULTS *
***********************

                WHEN CS-ROW-RESULT
                  MOVE LOW-VALUES TO A5PANELO
                  PERFORM RESULT-ROW-PROCESSING
                  MOVE 'Y' TO SW-FETCH
                  PERFORM FETCH-ROW-PROCESSING UNTIL NO-MORE-ROWS
```

```
*************************************************************
* PROCESS PARAMETER RESULTS - THERE SHOULD BE NO PARAMETERS *
* TO PROCESS                                                *
*************************************************************

               WHEN CS-PARAM-RESULT
                 MOVE 'Y' TO SW-FETCH

*************************************************************
* PROCESS STATUS RESULTS - THE STORED PROCEDURE STATUS RESULT *
* WILL NOT BE PROCESSED IN THIS EXAMPLE                      *
*************************************************************

               WHEN CS-STATUS-RESULT
                 MOVE 'Y' TO SW-FETCH

*************************************************************
* PRINT AN ERROR MESSAGE IF THE SERVER ENCOUNTERED AN ERROR *
* WHILE EXECUTING THE REQUEST                               *
*************************************************************

               WHEN CS-CMD-FAIL
                 STRING
                    'CTBRESUL returned CS-CMD-FAIL restype'
                    DELIMITED BY SIZE INTO MSGSTR
                 PERFORM PRINT-MSG

*****************************************************************
* PRINT A MESSAGE FOR SUCCESSFUL COMMANDS THAT RETURNED NO DATA *
* (OPTIONAL)                                                    *
*****************************************************************
               WHEN CS-CMD-SUCCEED
                 STRING
                    'CTBRESUL returned CS-CMD-SUCCEED restype'
                    DELIMITED BY SIZE INTO MSGSTR

*******************************************************
* PRINT A MESSAGE FOR REQUESTS THAT HAVE BEEN PROCESSED *
* SUCCESSFULLY (OPTIONAL)                               *
*******************************************************

               WHEN CS-CMD-DONE
                 STRING 'CTBRESUL returned CS-CMD-DONE restype'
                             DELIMITED BY SIZE INTO MSGSTR

               WHEN OTHER
```

```
                    STRING 'CTBRESUL returned UNKNOWN restype'
                              DELIMITED BY SIZE INTO MSGSTR
                    PERFORM PRINT-MSG
                    MOVE 'N' TO SW-RESULTS

                END-EVALUATE

*********************************************************
* PRINT AN ERROR MESSAGE IF THE CTBRESULTS CALL FAILED *
*********************************************************

            WHEN CS-FAIL
              MOVE 'N' TO SW-RESULTS
              STRING 'CTBRESUL returned CS-FAIL ret-code'
                        DELIMITED BY SIZE INTO MSGSTR
              PERFORM PRINT-MSG

************************************************************
* DROP OUT OF THE RESULTS LOOP IF NO MORE RESULT SETS ARE   *
* AVAILABLE FOR PROCESSING OR IF THE RESULTS WERE CANCELLED *
************************************************************

            WHEN CS-END-RESULTS
              MOVE 'N' TO SW-RESULTS

            WHEN CS-CANCELLED
              MOVE 'N' TO SW-RESULTS

            WHEN OTHER
              MOVE 'N' TO SW-RESULTS
              STRING 'CTBRESUL returned UNKNOWN ret-code'
                        DELIMITED BY SIZE INTO MSGSTR
              PERFORM PRINT-MSG

         END-EVALUATE.

         MOVE 0 TO RESTYPE.

     RESULTS-PROCESSING-EXIT.
         EXIT.

*=========================================================
*==                                                     ==
*== Subroutine to process result rows                   ==
*==                                                     ==
*=========================================================
```

```
 RESULT-ROW-PROCESSING.

     CALL 'CTBRESIN' USING CSL-CMD-HANDLE,
                           CSL-RC,
                           CS-NUMDATA,
                           RF-NUMDATA,
                           RF-NUMDATA-SIZE,
                           CF-COL-LEN.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBRESINFO failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

     COMPUTE FF-ROW-NUM = FF-ROW-NUM + 1.

*********************************
* display number of connections *
*********************************

     MOVE CF-MAXCONNECT   TO OR2-MAXCONNECT.
     MOVE OUTPUT-ROW-STR2 TO RSLTNO(FF-ROW-NUM).
     COMPUTE FF-ROW-NUM = FF-ROW-NUM + 2.

*********************************
* display the number of columns *
*********************************

     MOVE RF-NUMDATA      TO OR4-NUMDATA.
     MOVE OUTPUT-ROW-STR4 TO RSLTNO(FF-ROW-NUM).

     IF RF-NUMDATA NOT EQUAL 2
       THEN
         STRING 'CTBRESINFO returned wrong # of parms' DELIMITED
                                       BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

     COMPUTE FF-ROW-NUM = FF-ROW-NUM + 2.

**-------------------------------------------------------------
```

```
 **   Setup column headings
 **-------------------------------------------------------------

      MOVE 'FirstName    EducLvl' TO RSLTNO(FF-ROW-NUM).
      COMPUTE FF-ROW-NUM = FF-ROW-NUM + 1.
      MOVE '===========  =======' TO RSLTNO(FF-ROW-NUM).

      PERFORM BIND-COLUMNS
         VARYING I FROM 1 BY 1
            UNTIL I IS GREATER THAN RF-NUMDATA.

  RESULT-ROW-PROCESSING-EXIT.
      EXIT.

 *=========================================================
 *==                                                     ==
 *== Subroutine to bind each data                        ==
 *==                                                     ==
 *=========================================================
  BIND-COLUMNS.

       CALL 'CTBDESCR' USING CSL-CMD-HANDLE,
                             CSL-RC,
                             I,
                             DATAFMT.

       IF CSL-RC NOT EQUAL CS-SUCCEED
         THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBDESCR failed'
                  DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
       END-IF.

 **-------------------------------------------------------
 **   We need to bind the data to program variables.
 **   We don't care about the indicator variable
 **   so we'll pass NULL for that parameter in OC-BIND().
 **-------------------------------------------------------



*****************
 * ROWs per FETCH *
 *****************
```

```
MOVE 1 TO DF-COUNT

EVALUATE DF-DATATYPE

  WHEN CS-SMALLINT-TYPE

    CALL 'CTBBIND' USING CSL-CMD-HANDLE,
                         CSL-RC,
                         I,
                         DATAFMT,
                         DATA-SMALLINT,
                         CF-COL-LEN,
                         CS-PARAM-NOTNULL,
                         CF-COL-INDICATOR,
                         CS-PARAM-NULL

    IF CSL-RC NOT EQUAL CS-SUCCEED
      THEN
        MOVE SPACES TO MSGSTR
        STRING 'CTBBIND CS-SMALLINT-TYPE failed' DELIMITED
                                  BY SIZE INTO MSGSTR
        PERFORM PRINT-MSG
        PERFORM ALL-DONE
    END-IF

  WHEN CS-VARCHAR-TYPE

    MOVE LENGTH OF CF-COL-FIRSTNME-TXT TO DF-MAXLENGTH

    CALL 'CTBBIND' USING CSL-CMD-HANDLE,
                         CSL-RC,
                         I,
                         DATAFMT,
                         CF-COL-FIRSTNME,
                         CF-COL-LEN,
                         CS-PARAM-NOTNULL,
                         CF-COL-INDICATOR,
                         CS-PARAM-NULL

    IF CSL-RC NOT EQUAL CS-SUCCEED
      THEN
        MOVE SPACES TO MSGSTR
        STRING 'CTBBIND CS-VARCHAR-TYPE failed' DELIMITED
                                  BY SIZE INTO MSGSTR
        PERFORM PRINT-MSG
        PERFORM ALL-DONE
```

```
           END-IF.

 BIND-COLUMNS-EXIT.
     EXIT.

*=========================================================
*==                                                     ==
*== Subroutine to fetch row processing                  ==
*==                                                     ==
*=========================================================
 FETCH-ROW-PROCESSING.

     CALL 'CTBFETCH' USING CSL-CMD-HANDLE,
                           CSL-RC,
                           CS-UNUSED,
                           CS-UNUSED,
                           CS-UNUSED,
                           FF-ROWS-READ.

     EVALUATE CSL-RC

         WHEN CS-SUCCEED
              MOVE 'Y'              TO SW-FETCH
              MOVE CS-VARCHAR-TYPE TO DF-DATATYPE
              MOVE LENGTH OF CF-COL-FIRSTNME-TXT
                                    TO DF-MAXLENGTH
              MOVE CS-CHAR-TYPE     TO DF2-DATATYPE
              MOVE LENGTH OF CF-COL-FIRSTNME-CHAR
                                    TO DF2-MAXLENGTH

              CALL 'CSBCONVE' USING CSL-CTX-HANDLE,
                                    CSL-RC,
                                    DATAFMT,
                                    CF-COL-FIRSTNME,
                                    DATAFMT2,
                                    CF-COL-FIRSTNME-CHAR,
                                    CF-COL-LEN

              IF CSL-RC NOT EQUAL CS-SUCCEED
                THEN
                   MOVE SPACES TO MSGSTR
                   STRING 'CSBCONVERT CS-VARCHAR-TYPE failed'
                               DELIMITED BY SIZE INTO MSGSTR
                   PERFORM PRINT-MSG
                   PERFORM ALL-DONE
              END-IF
```

```
               COMPUTE FF-ROW-NUM = FF-ROW-NUM + 1

***********************************
* save ROW RESULTS for later display *
***********************************

               MOVE CF-COL-FIRSTNME-CHAR TO
                    OR-COL-FIRSTNME-CHAR
               MOVE DATA-SMALLINT TO
                    OR-COL-EDUCLVL

               IF FF-ROW-NUM > MAX-SCREEN-ROWS
                 THEN
                   STRING 'Please press return to continue.'
                          DELIMITED BY SIZE INTO MSG1O
                   MOVE SPACES TO MSG-TEXT-2
                   PERFORM DISP-DATA
                   PERFORM CLEAR-SCREEN-DATA
                           VARYING FF-ROW-NUM FROM 1 BY 1
                           UNTIL FF-ROW-NUM > MAX-SCREEN-ROWS
                   MOVE LOW-VALUES TO A5PANELO
                   COMPUTE PAGE-CNT = PAGE-CNT + 1
                   MOVE 1 TO FF-ROW-NUM
**-------------------------------------------------------------
**   Setup column headings
**-------------------------------------------------------------
                   MOVE 'FirstName    EducLvl' TO
                                    RSLTNO(FF-ROW-NUM)
                   COMPUTE FF-ROW-NUM = FF-ROW-NUM + 1
                   MOVE '===========  =======' TO
                                    RSLTNO(FF-ROW-NUM)
                   COMPUTE FF-ROW-NUM = FF-ROW-NUM + 1
               END-IF

               MOVE OUTPUT-ROW-STR TO RSLTNO(FF-ROW-NUM)

               MOVE SPACES         TO CF-COL-FIRSTNME-TXT

          WHEN CS-END-DATA
               MOVE SPACES     TO MSG1O
               MOVE 'N'        TO SW-FETCH
               MOVE 'Press Clear To Exit'
                               TO MSG-TEXT-2
               STRING 'All rows processing completed!'
                      DELIMITED BY SIZE INTO MSG1O
```

```
               PERFORM DISP-DATA

          WHEN CS-FAIL
               MOVE 'N'    TO SW-FETCH
               MOVE SPACES TO MSGSTR
               STRING 'CTBFETCH returned CS-FAIL ret-code'
                      DELIMITED BY SIZE INTO MSGSTR
               PERFORM PRINT-MSG

          WHEN CS-ROW-FAIL
               MOVE 'N'    TO SW-FETCH
               MOVE SPACES TO MSGSTR
               STRING 'CTBFETCH returned CS-ROW-FAIL ret-code'
                               DELIMITED BY SIZE INTO MSGSTR
               PERFORM PRINT-MSG

          WHEN CS-CANCELLED
               MOVE 'N'         TO SW-FETCH
               MOVE MF-CANCELED TO MSG1O
               PERFORM PRINT-MSG

          WHEN OTHER
               MOVE 'N'    TO SW-FETCH
               MOVE SPACES TO MSGSTR
               STRING 'CTBFETCH returned UNKNOWN ret-code'
                               DELIMITED BY SIZE INTO MSGSTR
               PERFORM PRINT-MSG

      END-EVALUATE.

 FETCH-ROW-PROCESSING-EXIT.
      EXIT.

*=========================================================
*==                                                     ==
*== Subroutine to display output                        ==
*==                                                     ==
*=========================================================
 DISP-DATA.

      MOVE TMP-DATE   TO SDATEO.
      MOVE TMP-TIME   TO STIMEO.
      MOVE 'SYCTSAA5' TO PROGNMO.
      MOVE PAGE-CNT   TO SPAGEO.

      MOVE DFHBMPRO   TO SERVERA.
```

```
     MOVE PF-SERVER  TO SERVERO.

     MOVE DFHBMPRO   TO USERA.
     MOVE PF-USER    TO USERO.

     MOVE DFHBMPRO   TO NETDRVA.
     MOVE PF-NETDRV  TO NETDRVO.

     MOVE DFHBMDAR   TO PSWDA.
     MOVE PF-PWD    TO PSWDO.
     MOVE MSG-TEXT-2 TO MSG2O.

*********************************************************
* PRINT ALL THE RETURNED ROWS FROM THE STORED PROCEDURE *
*********************************************************

*******************
* DISPLAY THE DATA *
*******************

*    EXEC CICS SEND MAP('SYCTBA5')
*                   MAPSET('SYCTBA5')
     EXEC CICS SEND MAP('A5PANEL')
                    MAPSET('SYCTBA5')
                    CURSOR
                    FRSET
                    ERASE
                    FREEKB
     END-EXEC.

     EXEC CICS RECEIVE INTO(QF-ANSWER)
                       LENGTH(QF-LEN)
                       MAXLENGTH(QF-MAXLEN)
                       RESP(CICS-RESPONSE)
     END-EXEC.

 DISP-DATA-EXIT.
     EXIT.

*=========================================================
*==                                                     ==
*== Subroutine to print output messages.                ==
*==                                                     ==
*=========================================================
 PRINT-MSG.
```

```
      MOVE LOW-VALUES TO A5PANELO.
      MOVE CSL-RC     TO SAMP-RC.
      MOVE RESTYPE    TO REST-TYPE.

      IF DIAG-MSGS-INITIALIZED
        THEN
          PERFORM GET-DIAG-MESSAGES
      END-IF.

***********************
* DISPLAY THE MESSAGE *
***********************

      MOVE DISP-MSG   TO MSG1O.

      IF NO-ERRORS
        THEN
          PERFORM DISP-DATA.

      MOVE C-Y    TO NO-ERRORS-SW.
      MOVE SPACES TO MSGSTR.
      MOVE SPACES TO MSG1O.
      MOVE ZERO   TO SAMP-RC.
      MOVE ZERO   TO REST-TYPE.

 PRINT-MSG-EXIT.
      EXIT.

*============================================================
*==                                                        ==
*== Subroutine to drop and to deallocate all handlers,     ==
*== to close server connection and exit client library     ==
*==                                                        ==
*============================================================
 ALL-DONE.

      PERFORM CLOSE-CONNECTION.
      PERFORM QUIT-CLIENT-LIBRARY.
      STOP RUN.

 ALL-DONE-EXIT.
      EXIT.

*================================================================
*==                                                            ==
*== Subroutine to perform drop command handler, close          ==
```

```
*== server connection, and deallocate Connection Handler. ==
*==                                                        ==
*============================================================
 CLOSE-CONNECTION.

**************************
* DROP THE COMMAND HANDLE *
**************************

     CALL 'CTBCMDDR' USING CSL-CMD-HANDLE
                           CSL-RC.

     IF CSL-RC = CS-FAIL
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBCMDDR failed' DELIMITED BY
                 SIZE INTO MSGSTR
         PERFORM PRINT-MSG
     END-IF.

******************************
* CLOSE THE SERVER CONNECTION *
******************************

     CALL 'CTBCLOSE' USING CSL-CON-HANDLE
                           CSL-RC
                           CS-UNUSED.

     IF CSL-RC = CS-FAIL
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBCLOSE failed' DELIMITED BY
                 SIZE INTO MSGSTR
         PERFORM PRINT-MSG
     END-IF.

***********************************
* DE-ALLOCATE THE CONNECTION HANDLE *
***********************************

     CALL 'CTBCONDR' USING CSL-CON-HANDLE
                           CSL-RC.

     IF CSL-RC = CS-FAIL
       THEN
         MOVE SPACES TO MSGSTR
```

```
          STRING 'CTBCONDR failed' DELIMITED BY
                  SIZE INTO MSGSTR
          PERFORM PRINT-MSG
      END-IF.

 CLOSE-CONNECTION-EXIT.
      EXIT.

*================================================
*==                                            ==
*== Subroutine to perform exit client library and ==
*== deallocate context structure.              ==
*==                                            ==
*================================================
 QUIT-CLIENT-LIBRARY.

**************************
* EXIT THE CLIENT LIBRARY *
**************************

      CALL 'CTBEXIT' USING CSL-CTX-HANDLE
                           CSL-RC
                           CS-UNUSED.
      IF CSL-RC = CS-FAIL
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBEXIT failed' DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
      END-IF.

***********************************
* DE-ALLOCATE THE CONTEXT STRUCTURE *
***********************************

      CALL 'CSBCTXDR' USING CSL-CTX-HANDLE
                            CSL-RC.

      IF CSL-RC = CS-FAIL
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CSBCTXDR failed' DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
      END-IF.

      EXEC CICS RETURN END-EXEC.
```

```
 QUIT-CLIENT-LIBRARY-EXIT.
     EXIT.

*=====================================================
*==                                                 ==
*== Subroutine to retrieve any diagnostic messages ==
*==                                                 ==
*=====================================================
 GET-DIAG-MESSAGES.

***********************************
* Disable calls to this subroutine *
***********************************

     MOVE 'N' TO SW-DIAG.

*****************************
* First, get client messages *
*****************************

     CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                          CSL-RC,
                          CS-UNUSED,
                          CS-STATUS,
                          CS-CLIENTMSG-TYPE,
                          CS-UNUSED,
                          DG-NUM-OF-MSGS.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBDIAG CS-STATUS CS-CLIENTMSG-TYP fail'
                         DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
       ELSE
         IF DG-NUM-OF-MSGS > 0
           THEN
             PERFORM RETRIEVE-CLIENT-MSGS
                 VARYING I FROM 1 BY 1
                     UNTIL I IS GREATER THAN DG-NUM-OF-MSGS
         END-IF
     END-IF.
```

```
*****************************
 * Then, get server messages *
 *****************************

      CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                           CSL-RC,
                           CS-UNUSED,
                           CS-STATUS,
                           CS-SERVERMSG-TYPE,
                           CS-UNUSED,
                           DG-NUM-OF-MSGS.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          STRING 'CTBDIAG CS-STATUS CS-SERVERMSG-TYP fail'
                         DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
        ELSE
          IF DG-NUM-OF-MSGS > 0
            THEN
              PERFORM RETRIEVE-SERVER-MSGS
                  VARYING I FROM 1 BY 1
                      UNTIL I IS GREATER THAN DG-NUM-OF-MSGS
          END-IF
      END-IF.

  GET-DIAG-MESSAGES-EXIT.
      EXIT.

 *============================================================
 *==                                                        ==
 *== Subroutine to retrieve diagnostic messages from client ==
 *==                                                        ==
 *============================================================
  RETRIEVE-CLIENT-MSGS.

      MOVE 1 TO I1.

      CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                           CSL-RC,
                           CS-UNUSED,
                           CS-GET,
                           CS-CLIENTMSG-TYPE,
                           DG-MSGNO,
                           CLIENT-MSG.
```

```
      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBDIAG CS-GET CS-CLIENTMSG-TYPE failed'
                          DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.

************************
* display message text *
************************

      MOVE DISP-CLIENT-MSG-HDR TO RSLTNO( I1 ).
      MOVE 3 TO I1.

      MOVE CM-SEVERITY       TO CM-SEVERITY-DATA.
      MOVE CM-STATUS         TO CM-STATUS-DATA.
      MOVE DISP-CLIENT-MSG-1 TO RSLTNO( I1 ).
      COMPUTE I1 EQUAL I1 + 1

      MOVE CM-MSGNO          TO CM-OC-MSGNO-DATA.
      MOVE DISP-CLIENT-MSG-2 TO RSLTNO( I1 ).
      COMPUTE I1 EQUAL I1 + 1

      IF CM-MSGNO NOT EQUAL 0
        THEN
          MOVE SPACES            TO CM-OC-MSG-DATA
          MOVE CM-TEXT           TO CM-OC-MSG-DATA
          MOVE CM-TEXT           TO DISP-CLIENT-MSG-3A
          MOVE DISP-CLIENT-MSG-3 TO RSLTNO( I1 )
          COMPUTE I1 EQUAL I1 + 1
          IF CM-TEXT-LEN > 66
            THEN
              MOVE CM-OC-MSG-DATA-2   TO CM-OC-MSG-DATA-X
              MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
              COMPUTE I1 EQUAL I1 + 1
              IF CM-TEXT-LEN > 132
                THEN
                  MOVE SPACES             TO CM-OC-MSG-DATA-X
                  MOVE CM-OC-MSG-DATA-3   TO CM-OC-MSG-DATA-X
                  MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
                  COMPUTE I1 EQUAL I1 + 1
                  IF CM-TEXT-LEN > 198
                    THEN
```

```
                MOVE SPACES             TO CM-OC-MSG-DATA-X
                MOVE CM-OC-MSG-DATA-4   TO CM-OC-MSG-DATA-X
                MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
                COMPUTE I1 EQUAL I1 + 1
            END-IF
        END-IF
      END-IF
  ELSE
    MOVE DISP-EMPTY-CLIENT-MSG-3 TO RSLTNO( I1 )
    COMPUTE I1 EQUAL I1 + 1
END-IF.

MOVE CM-OS-MSGNO       TO CM-OS-MSGNO-DATA.
MOVE DISP-CLIENT-MSG-4 TO RSLTNO( I1 ).
COMPUTE I1 EQUAL I1 + 1

IF CM-OS-MSGNO NOT EQUAL 0
  THEN
    MOVE SPACES            TO CM-OS-MSG-DATA
    MOVE CM-OS-MSGTXT      TO CM-OS-MSG-DATA
    MOVE SPACES            TO DISP-CLIENT-MSG-5A
    MOVE CM-OS-MSGTXT      TO DISP-CLIENT-MSG-5A
    MOVE DISP-CLIENT-MSG-5 TO RSLTNO( I1 )
    COMPUTE I1 EQUAL I1 + 1
    IF CM-OS-MSGTEXT-LEN > 66
      THEN
        MOVE SPACES             TO CM-OC-MSG-DATA-X
        MOVE CM-OS-MSG-DATA-2   TO CM-OC-MSG-DATA-X
        MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
        COMPUTE I1 EQUAL I1 + 1
        IF CM-OS-MSGTEXT-LEN > 132
          THEN
            MOVE SPACES             TO CM-OC-MSG-DATA-X
            MOVE CM-OS-MSG-DATA-3   TO CM-OC-MSG-DATA-X
            MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
            COMPUTE I1 EQUAL I1 + 1
            IF CM-OS-MSGTEXT-LEN > 198
              THEN
                MOVE SPACES             TO CM-OC-MSG-DATA-X
                MOVE CM-OS-MSG-DATA-4   TO CM-OC-MSG-DATA-X
                MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
                COMPUTE I1 EQUAL I1 + 1
            END-IF
        END-IF
    END-IF
  ELSE
```

```
           MOVE DISP-EMPTY-CLIENT-MSG-5 TO RSLTNO( I1 )
           COMPUTE I1 EQUAL I1 + 1
       END-IF.

 RETRIEVE-CLIENT-MSGS-EXIT.
       EXIT.

*============================================================
*==                                                        ==
*== Subroutine to retrieve diagnostic messages from server ==
*==                                                        ==
*============================================================
 RETRIEVE-SERVER-MSGS.

       CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                            CSL-RC,
                            CS-UNUSED,
                            CS-GET,
                            CS-SERVERMSG-TYPE,
                            DG-MSGNO,
                            SERVER-MSG.

       IF CSL-RC NOT EQUAL CS-SUCCEED
         THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBDIAG CS-GET CS-SERVERMSG-TYPE failed'
                           DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
       END-IF.

************************
* display message text *
************************

       MOVE SM-MSGNO   TO SM-MSG-NO-DATA.
       MOVE SM-SEV     TO SM-SEVERITY-DATA.
       MOVE SM-STATE   TO SM-STATE-DATA.

       MOVE SM-LINE    TO SM-LINE-NO-DATA.
       MOVE SM-STATUS  TO SM-STATUS-DATA.

       MOVE SPACES     TO SM-SVRNAME-DATA.
       MOVE SM-SVRNAME TO SM-SVRNAME-DATA.

       MOVE SPACES     TO SM-PROC-ID-DATA.
```

```
      MOVE SM-PROC    TO SM-PROC-ID-DATA.

      MOVE SPACES     TO SM-MSG-DATA.
      MOVE SM-TEXT    TO SM-MSG-DATA.

      MOVE SPACES     TO DISP-SERVER-MSG-5A.
      MOVE SM-TEXT    TO DISP-SERVER-MSG-5A.


      MOVE DISP-SERVER-MSG-HDR TO RSLTNO (1).
      MOVE DISP-SERVER-MSG-1   TO RSLTNO (3).
      MOVE DISP-SERVER-MSG-2   TO RSLTNO (4).
      MOVE DISP-SERVER-MSG-3   TO RSLTNO (5).
      MOVE DISP-SERVER-MSG-4   TO RSLTNO (6).

      MOVE DISP-SERVER-MSG-5   TO RSLTNO (7).
      IF SM-TEXT-LEN > 66
        THEN
          MOVE SPACES             TO SM-MSG-DATA-X
          MOVE SM-MSG-DATA-2      TO SM-MSG-DATA-X
          MOVE DISP-SERVER-MSG-5X TO RSLTNO(8)
          IF SM-TEXT-LEN > 132
            THEN
              MOVE SPACES             TO SM-MSG-DATA-X
              MOVE SM-MSG-DATA-3      TO SM-MSG-DATA-X
              MOVE DISP-SERVER-MSG-5X TO RSLTNO(9)
              IF SM-TEXT-LEN > 198
                THEN
                  MOVE SPACES             TO SM-MSG-DATA-X
                  MOVE SM-MSG-DATA-4      TO SM-MSG-DATA-X
                  MOVE DISP-SERVER-MSG-5X TO RSLTNO(10)
              END-IF
          END-IF
      END-IF.

 RETRIEVE-SERVER-MSGS-EXIT.
      EXIT.

*============================================================
*==                                                        ==
*== Subroutine to clear the output screen                  ==
*==                                                        ==
*============================================================
 CLEAR-SCREEN-DATA.

      MOVE SPACES TO RSLTNO( FF-ROW-NUM ).
```

```
 CLEAR-SCREEN-DATA-EXIT.

      EXIT.

*=============================================================
*==                                                         ==
*== Subroutine to handle MAPFAIL condition                  ==
*==                                                         ==
*=============================================================
 NO-INPUT.
*---------

      MOVE 'Please Enter Input Fields' TO MSG-TEXT-1.

      GO TO GET-INPUT-AGAIN.

*=============================================================
*==                                                         ==
*== Subroutine to handle AID condition                      ==
*==                                                         ==
*=============================================================
 GETOUT.
*--------

      EXEC CICS RETURN END-EXEC.

      STOP RUN.



*=============================================================
 *==                                                         ==
 *== Subroutine to handle ERROR condition                    ==
 *==                                                         ==
 *=============================================================
  ERRORS.
 *--------

      EXEC CICS DUMP DUMPCODE('ERRS') END-EXEC.

      STOP RUN.
```

SYCTSAP5 - sample language request

The purpose of this sample program is to demonstrate:

- Explicit conversion of a CS-DECIMAL type column on the server to IBM packed decimal
- Explicit conversion of a CS-DATETIME column on the server to CHAR DATE format
- Explicit conversion of a VARCHAR datatype to CHAR datatype.

This sample program retrieves information from the table SYBASE.NEWTABLE on the target server.

```
            *@(#) syctsap5.cobol 1.2 4/9/96        */


*****************************************************************
*
* Confidential property of Sybase, Inc.
* (c) Copyright Sybase, Inc. 1985 TO 1997.
* All rights reserved.
*

*****************************************************************

******* SYCTSAP5 - Client Language Request APPL - COBOL - CICS **
**
**  CICS TRANID:  SYP5
**
**  PROGRAM:  SYCTSAP5
**
**  PURPOSE:  Demonstrates Open Client for CICS CALLs.
**
**  FUNCTION: Illustrates how to send a language request with
**            parameters to:
**
**              - A SQL Server
**
**            Illustrates the explicit conversion of:
**            VARCHAR to CHAR data type
**            DECIMAL to PACKED DECIMAL  data type
**            DATETIME to CHAR data type
**
**            SQL Server:
**
**              If the request is sent to a SQL Server it
```

```
**              executes the SQL statement:
**
**                SELECT  FIRSTNME, MILAGE, SERVICEDATE
**                        FROM  SYBASE.NEWTABLE
**
**              Note: The Net-Gateway/MCG product includes a script
**                    that creates this procedure in a target SQL
**                    server.
**
**  PREREQS:  Before running SYCTSAP5, make sure that the server
**            you wish to access has an entry in the Connection
**            Router Table for that Server and the MCG(s) that
**            you wish to use.
**
**  INPUT:    On the input screen, make sure to enter the Server
**            name, user id, and password for the target server.
**            TRAN NAME is not used for LAN servers.
**
**  Open Client CALLs used in this sample:
**
**    CSBCONVERT    convert a datatype from one value to another
**    CSBCTXALLOC   allocate a context
**    CSBCTXDROP    drop a context
**    CTBBIND       bind a column variable
**    CTBCLOSE      close a server connection
**    CTBCONFIG     set or retrieve context properties
**    CTBCMDALLOC   allocate a command
**    CTBCMDDROP    drop a command
**    CTBCOMMAND    initiate remote procedure CALL
**    CTBCONALLOC   allocate a connection
**    CTBCONDROP    drop a connection
**    CTBCONPROPS   alter properties of a connection
**    CTBCONNECT    open a server connection
**    CTBDESCRIBE   return a description of RESULT data
**    CTBDIAG       retrieve SQLCODE messages
**    CTBEXIT       exit client library
**    CTBFETCH      FETCH RESULT data
**    CTBINIT       init client library
**    CTBPARAM      define a command PARAMETER
**    CTBRESULTS    set up RESULT data
**    CTBRESINFO    return RESULT set info
**    CTBSEND       send a request TO the server
**
*****************************************************************

 IDENTIFICATION DIVISION.
```

```
 PROGRAM-ID.  SYCTSAP5.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER.  xyz.
 OBJECT-COMPUTER.  xyz.
 DATA DIVISION.
 WORKING-STORAGE SECTION.

*****************************************************************
** CLIENT LIBRARY COBOL COPY BOOK
*****************************************************************
   COPY CTPUBLIC.

*****************************************************************
** CICS BMS DEFINITIONS
*****************************************************************

   COPY SYCTBA5.

*****************************************************************
*  Standard CICS Attribute and Print Control Chararcter List
*****************************************************************

   COPY DFHBMSCA.

*****************************************************************
** CICS Standard Attention Identifiers Cobol Copy Book
*****************************************************************

   COPY DFHAID.

*****************************************************************
*    CONSTANTS
*****************************************************************
 01  C-N                        PIC X(01) VALUE 'N'.
 01  C-Y                        PIC X(01) VALUE 'Y'.
 01  I1                         PIC S9(9) COMP SYNC VALUE IS 0.
 01  I2                         PIC S9(9) COMP SYNC VALUE IS 0.

 01  MSG-TEXT-1                 PIC X(70) VALUE ' '.
 01  MSG-TEXT-2                 PIC X(70)
                                VALUE 'Press Clear To Exit'.
 01  PAGE-CNT                   PIC S9(4) COMP VALUE +0.
 01  UTIME                      PIC S9(15) COMP-3.
 01  TMP-DATE                   PIC X(08).
 01  TMP-TIME                   PIC X(08).
```

```
 01  MAX-SCREEN-ROWS             PIC S9(4) VALUE +10.

 01  ENTER-DATA-SW               PIC X(01) VALUE 'N'.

*****************************************************************
*    OPEN CLIENT VARIABLES
*****************************************************************
 01  STRLEN                      PIC S9(9) COMP VALUE +0.
 01  OUTLEN                      PIC S9(9) COMP VALUE +0.
 01  RESTYPE                     PIC S9(9) COMP VALUE +0.
 01  NETDRIVER                   PIC S9(9) COMP VALUE +9999.

**---------------------------------------------------------------
** WORK AREAS
**---------------------------------------------------------------
 01  NO-MORE-MSGS-SW             PIC X(01).
     88  NO-MORE-MSGS VALUE 'Y'.

 01  NO-ERRORS-SW                PIC X(01).
     88  NO-ERRORS    VALUE 'N'.

 01  SWITCHES.
     05  SW-RESULTS                PIC X(01) VALUE 'Y'.
         88  NO-MORE-RESULTS VALUE 'N'.
     05  SW-FETCH                  PIC X(01) VALUE 'Y'.
         88  NO-MORE-ROWS VALUE 'N'.
     05  SW-DIAG                   PIC X(01) VALUE 'N'.
         88  DIAG-MSGS-INITIALIZED VALUE 'Y'.

 01  INTERNAL-FIELDS.
     05  I                     PIC S9(9) COMP.
     05  CF-FOUR               PIC S9(9) COMP VALUE +4.
     05  CF-LANG2-SIZE         PIC S9(9) COMP VALUE +85.
     05  DATA-PACKED370        PIC S9(15)V9(3) COMP-3 VALUE +0.

 01  CS-LIB-MISC-FIELDS.
     05  CSL-CMD-HANDLE        PIC S9(9) COMP VALUE +0.
     05  CSL-CON-HANDLE        PIC S9(9) COMP VALUE +0.
     05  CSL-CTX-HANDLE        PIC S9(9) COMP VALUE +0.
     05  CSL-RC                PIC S9(9) COMP VALUE +0.

 01  PROPS-FIELDS.
     05  PF-SERVER             PIC X(30) VALUE IS SPACES.
     05  PF-SERVER-SIZE        PIC S9(9) COMP VALUE +0.
     05  PF-USER               PIC X(08) VALUE IS SPACES.
```

```
    05  PF-USER-SIZE          PIC S9(9) COMP VALUE +0.
    05  PF-PWD                PIC X(08) VALUE IS SPACES.
    05  PF-PWD-SIZE           PIC S9(9) COMP VALUE +0.
    05  PF-TRAN               PIC X(08) VALUE IS SPACES.
    05  PF-TRAN-SIZE          PIC S9(9) COMP VALUE +0.
    05  PF-NETDRV             PIC X(08) VALUE IS SPACES.
    05  PF-DRV-SIZE           PIC S9(9) COMP VALUE +0.
    05  PF-STRLEN             PIC S9(9) COMP.
    05  PF-MSGLIMIT           PIC S9(9) COMP.

01  DIAG-FIELDS.
    05  DG-MSGNO              PIC S9(9) COMP VALUE +1.
    05  DG-NUM-OF-MSGS        PIC S9(9) COMP VALUE +0.

01  CONFIG-FIELDS.
    05  CF-MAXCONNECT         PIC S9(9) COMP.
    05  CF-OUTLEN             PIC S9(9) COMP.

01  FETCH-FIELDS.
    05  FF-ROWS-READ          PIC S9(9) COMP.
    05  FF-ROW-NUM            PIC S9(9) COMP VALUE +0.

01  RESINFO-FIELDS.
    05  RF-NUMDATA            PIC S9(9) COMP.
    05  RF-NUMDATA-SIZE       PIC S9(9) COMP VALUE +4.

01  OUTPUT-ROW.
    05  OR-COL-FIRSTNME-CHAR  PIC X(12).
    05  SPACE1                PIC X(01) VALUE ' '.
    05  OR-COL-MILAGE         PIC -9(16).9(2) VALUE '+0'.
    05  SPACE1                PIC X(12) VALUE ' '.
    05  OR-COL-SERVICEDATE PIC X(25) VALUE ' '.

01  OUTPUT-ROW-STR REDEFINES OUTPUT-ROW PIC X(70).

01  OUTPUT-ROW-SIZE           PIC S9(4) COMP VALUE +70.

01  OUTPUT-ROW2.
    05  OR2-MESG              PIC X(37)
                  VALUE 'The maximum number of connections is '.
    05  OR2-MAXCONNECT        PIC ZZZZ9.
    05  OR2-PERIOD            PIC X(01) VALUE '.'.

01  OUTPUT-ROW-STR2 REDEFINES OUTPUT-ROW2 PIC X(43).

01  OUTPUT-ROW2-SIZE          PIC S9(4) COMP VALUE +43.
```

```
 01  OUTPUT-ROW4.
     05  OR4-MESG              PIC X(25)
                               VALUE 'The number of columns is '.
     05  OR4-NUMDATA           PIC ZZZZ9.
     05  OR4-PERIOD            PIC X(01)     VALUE '.'.

 01  OUTPUT-ROW-STR4 REDEFINES OUTPUT-ROW4 PIC X(31).

 01  OUTPUT-ROW4-SIZE          PIC S9(4) COMP VALUE +31.

 01  COLUMN-FIELDS.
     05  CF-COL-FIRSTNME.
         10 CF-COL-FIRSTNME-LL PIC S9(9) COMP.
         10 CF-COL-FIRSTNME-TXT PIC X(12).
     05  CF-COL-FIRSTNME-CHAR  PIC X(12).
     05  CF-COL-MILAGE.
         10 CF-COL-MILAGE-PRECISION PIC X(1) VALUE ' '.
         10 CF-COL-MILAGE-SCALE PIC X(1) VALUE ' '.
         10 CF-COL-MILAGE-NUMBER PIC X(31).
     05  CF-COL-MILAGE-DECFORM PIC S9(15)V9(3) COMP-3   VALUE 0.
     05  CF-COL-MILAGE-CHAR PIC X(31) VALUE ' '.
     05  CF-COL-SERVICEDATE.
         10 CF-COL-DATE        PIC S9(4) COMP VALUE 0 .
         10 CF-COL-TIME        PIC S9(4) COMP VALUE 0.
     05  CF-COL-SERVICEDATE-BOUND PIC X(25) VALUE ' '.
     05  CF-COL-SERVICEDATE-CHAR PIC X(25) VALUE ' '.
     05  CF-COL-LEN            PIC S9(9) COMP VALUE 0.
     05  CF-COL-NULL           PIC S9(9) COMP VALUE +0.
     05  CF-COL-NUMBER         PIC S9(9) COMP VALUE +1.
     05  CF-COL-INDICATOR      PIC S9(4) COMP VALUE +0.

 01  LANG-FIELDS.
     05  CF-LANG1              PIC X(20)
         VALUE 'Wrong SQL statement'.
     05  CF-LANG2              PIC X(85)
         VALUE 'SELECT PLANEID , MILAGE,SERVICEDATE=DATEADD(DAY,20
-    ',SERVICEDATE) FROM SYBASE.NEWTABLE'.
     05  filler                PIC X(01) VALUE LOW-VALUE.

 01  MSG-FIELDS.
     05  MF-CANCELED           PIC X(16)
                               VALUE 'Cancel requested'.
     05  MF-CANCELED-SIZE      PIC S9(9) COMP VALUE +16.

 01  DATAFMT.
```

```
      05  DF-NAME                PIC X(132).
      05  DF-NAMELEN             PIC S9(9) COMP.
      05  DF-DATATYPE            PIC S9(9) COMP.
      05  DF-FORMAT              PIC S9(9) COMP.
      05  DF-MAXLENGTH           PIC S9(9) COMP.
      05  DF-SCALE               PIC S9(9) COMP VALUE 15.
      05  DF-PRECISION           PIC S9(9) COMP VALUE 31.
      05  DF-STATUS              PIC S9(9) COMP.
      05  DF-COUNT               PIC S9(9) COMP.
      05  DF-USERTYPE            PIC S9(9) COMP.
      05  DF-LOCALE              PIC X(68).

  01  DATAFMT2.
      05  DF2-NAME               PIC X(132).
      05  DF2-NAMELEN            PIC S9(9) COMP.
      05  DF2-DATATYPE           PIC S9(9) COMP.
      05  DF2-FORMAT             PIC S9(9) COMP.
      05  DF2-MAXLENGTH          PIC S9(9) COMP.
      05  DF2-SCALE              PIC S9(9) COMP VALUE 3.
      05  DF2-PRECISION          PIC S9(9) COMP VALUE 18.
      05  DF2-STATUS             PIC S9(9) COMP.
      05  DF2-COUNT              PIC S9(9) COMP.
      05  DF2-USERTYPE           PIC S9(9) COMP.
      05  DF2-LOCALE             PIC X(68).

  01 DISP-MSG.
     05 TEST-CASE                PIC X(08) VALUE IS 'SYCTSAP5'.
     05 FILLER                   PIC X(01) VALUE IS SPACES.
     05 MSG.
        10 SAMP-LIT              PIC X(05) VALUE IS 'rc = '.
        10 SAMP-RC               PIC -Z9.
        10 FILLER                PIC X(02) VALUE IS ', '.
        10 REST-LIT              PIC X(12) VALUE IS
                                    'Result Type:'.
        10 REST-TYPE             PIC Z(3)9.
        10 FILLER                PIC X(03) VALUE IS SPACES.
        10 MSGSTR                PIC X(40) VALUE IS SPACES.

  01  DISP-MSG-LEN               PIC S9(4) COMP  VALUE IS 65.
  01  MSG-LEN VALUE +0           PIC S9(4) COMP .

*******************************
** Client Message Structure  **
*******************************

  01  CLIENT-MSG.
```

```
    05  CM-SEVERITY           PIC S9(9) COMP SYNC.
    05  CM-MSGNO              PIC S9(9) COMP SYNC.
    05  CM-TEXT               PIC X(256).
    05  CM-TEXT-LEN           PIC S9(9) COMP SYNC.
    05  CM-OS-MSGNO           PIC S9(9) COMP SYNC.
    05  CM-OS-MSGTXT          PIC X(256).
    05  CM-OS-MSGTEXT-LEN     PIC S9(9) COMP SYNC.
    05  CM-STATUS             PIC S9(9) COMP.

01  DISP-CLIENT-MSG-HDR.
    05  CLIENT-MSG-HDR        PIC X(15) VALUE IS
                                  'Client Message:'.

01  DISP-CLIENT-MSG-1.
    05  FILLER                PIC X(02) VALUE IS SPACES.
    05  CM-SEVERITY-HDR       PIC X(09) VALUE IS 'Severity:'.
    05  FILLER                PIC X(02) VALUE IS SPACES.
    05  CM-SEVERITY-DATA      PIC Z(8)9.
    05  CM-STATUS-HDR         PIC X(12) VALUE IS
                                  ',  Status:  '.
    05  FILLER                PIC X(02) VALUE IS SPACES.
    05  CM-STATUS-DATA        PIC Z(8)9.

01  DISP-CLIENT-MSG-2.
    05  FILLER                PIC X(02) VALUE IS SPACES.
    05  CM-OC-MSGNO-HDR       PIC X(09) VALUE IS 'OC MsgNo:'.
    05  FILLER                PIC X(02) VALUE IS SPACES.
    05  CM-OC-MSGNO-DATA      PIC Z(8)9.

01  DISP-CLIENT-MSG-3.
    05  FILLER                PIC X(02) VALUE IS SPACES.
    05  CM-OC-MSG-HDR         PIC X(09) VALUE IS 'OC MsgTx:'.
    05  FILLER                PIC X(02) VALUE IS SPACES.
    05  CM-OC-MSG-DATA        PIC X(66).

01  DISP-CLIENT-MSG-3A.
    05  CM-OC-MSG-DATA-1      PIC X(66).
    05  CM-OC-MSG-DATA-2      PIC X(66).
    05  CM-OC-MSG-DATA-3      PIC X(66).
    05  CM-OC-MSG-DATA-4      PIC X(58).

01  DISP-CLIENT-MSG-3B.
    05  FILLER                PIC X(13) VALUE IS SPACES.
    05  CM-OC-MSG-DATA-X      PIC X(66).

01  DISP-EMPTY-CLIENT-MSG-3.
```

```
     05  FILLER                 PIC X(02) VALUE IS SPACES.
     05  CM-OC-MSG-HDR          PIC X(09) VALUE IS 'OC MsgTx:'.
     05  FILLER                 PIC X(02) VALUE IS SPACES.
     05  NO-DATA                PIC X(11) VALUE IS 'No Message!'.

 01  DISP-CLIENT-MSG-4.
     05  FILLER                 PIC X(02) VALUE IS SPACES.
     05  CM-OS-MSG-HDR          PIC X(09) VALUE IS 'OS MsgNo:'.
     05  FILLER                 PIC X(02) VALUE IS SPACES.
     05  CM-OS-MSGNO-DATA       PIC Z(8)9.

 01  DISP-CLIENT-MSG-5.
     05  FILLER                 PIC X(02) VALUE IS SPACES.
     05  CM-OS-MSG-HDR          PIC X(09) VALUE IS 'OS MsgTx:'.
     05  FILLER                 PIC X(02) VALUE IS SPACES.
     05  CM-OS-MSG-DATA         PIC X(66).

 01  DISP-CLIENT-MSG-5A.
     05  CM-OS-MSG-DATA-1       PIC X(66).
     05  CM-OS-MSG-DATA-2       PIC X(66).
     05  CM-OS-MSG-DATA-3       PIC X(66).
     05  CM-OS-MSG-DATA-4       PIC X(58).

 01  DISP-EMPTY-CLIENT-MSG-5.
     05  FILLER                 PIC X(02) VALUE IS SPACES.
     05  CM-OS-MSG-HDR          PIC X(09) VALUE IS 'OS MsgTx:'.
     05  FILLER                 PIC X(02) VALUE IS SPACES.
     05  NO-DATA                PIC X(11) VALUE IS 'No Message!'.

*******************************
** Server Message Structure  **
*******************************

 01  SERVER-MSG.
     05  SM-MSGNO               PIC S9(9) COMP.
     05  SM-STATE               PIC S9(9) COMP.
     05  SM-SEV                 PIC S9(9) COMP.
     05  SM-TEXT                PIC X(256).
     05  SM-TEXT-LEN            PIC S9(9) COMP.
     05  SM-SVRNAME             PIC X(256).
     05  SM-SVRNAME-LEN         PIC S9(9) COMP.
     05  SM-PROC                PIC X(256).
     05  SM-PROC-LEN            PIC S9(9) COMP.
     05  SM-LINE                PIC S9(9) COMP.
     05  SM-STATUS              PIC S9(9) COMP.
```

```
01  DISP-SERVER-MSG-HDR.
    05  SERVER-MSG-HDR         PIC X(15) VALUE IS
                                    'Server Message:'.

01  DISP-SERVER-MSG-1.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-MSG-NO-HDR          PIC X(09) VALUE IS
                                    'Message#:'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-MSG-NO-DATA         PIC Z(8)9.
    05  SM-SEVERITY-HDR        PIC X(12) VALUE IS
                                    ',  Severity:'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-SEVERITY-DATA       PIC Z(8)9.
    05  SM-STATE-HDR           PIC X(12) VALUE IS
                                    ',  State No:'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-STATE-DATA          PIC Z(8)9.

01  DISP-SERVER-MSG-2.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-LINE-NO-HDR         PIC X(09) VALUE IS
                                    'Line  No:'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-LINE-NO-DATA        PIC Z(8)9.
    05  SM-STATUS-HDR          PIC X(12) VALUE IS
                                    ',  Status  :'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-STATUS-DATA         PIC Z(8)9.

01  DISP-SERVER-MSG-3.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-SVRNAME-HDR         PIC X(09) VALUE IS 'Serv Nam:'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-SVRNAME-DATA        PIC X(66).
    05  FILLER                 PIC X(03) VALUE IS '...'.

01  DISP-SERVER-MSG-4.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-PROC-ID-HDR         PIC X(09) VALUE IS 'Proc  ID:'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-PROC-ID-DATA        PIC X(66).

01  DISP-SERVER-MSG-5.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-MSG-HDR             PIC X(09) VALUE IS 'Message :'.
```

```
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  SM-MSG-DATA           PIC X(66).

 01  DISP-SERVER-MSG-5A.
     05  SM-MSG-DATA-1         PIC X(66).
     05  SM-MSG-DATA-2         PIC X(66).
     05  SM-MSG-DATA-3         PIC X(66).
     05  SM-MSG-DATA-4         PIC X(58).

 01  DISP-SERVER-MSG-5X.
     05  FILLER                PIC X(13) VALUE IS SPACES.
     05  SM-MSG-DATA-X         PIC X(66).

 01  CICS-FIELDS.
     05  CICS-RESPONSE         PIC S9(9) COMP.

 01  QUERY-FIELDS.
     05  QF-LEN                PIC S9(4) COMP VALUE +1.
     05  QF-MAXLEN             PIC S9(4) COMP VALUE +1.
     05  QF-ANSWER             PIC X(01) VALUE IS SPACES.

 PROCEDURE DIVISION.
**********************

*************************
* CICS Condition Handler *
*************************

     EXEC CICS HANDLE CONDITION MAPFAIL(NO-INPUT)
                                ERROR(ERRORS)
     END-EXEC.

********************
* CICS Aid Handler *
********************

     EXEC CICS HANDLE AID ANYKEY(NO-INPUT)
                          CLEAR(GETOUT)
     END-EXEC.

*****************************
*    PROGRAM INITIALIZATION *
*****************************

     MOVE ZERO       TO RESTYPE CSL-RC.
```

```
      MOVE C-N    TO NO-MORE-MSGS-SW.
      MOVE C-N    TO NO-ERRORS-SW.
      MOVE C-Y    TO SW-DIAG.

      MOVE LOW-VALUES TO A5PANELO.
      MOVE -1         TO SERVERL.

      COMPUTE PAGE-CNT = PAGE-CNT + 1.

      PERFORM GET-SYSTEM-TIME.

 GET-INPUT-AGAIN.

      PERFORM DISPLAY-INITIAL-SCREEN.

      PERFORM GET-INPUT-DATA.

*********************************
*    ALLOCATE A CONTEXT STRUCTURE *
*********************************

      MOVE ZERO TO CSL-CTX-HANDLE.

      CALL 'CSBCTXAL' USING CS-VERSION-50
                            CSL-RC
                            CSL-CTX-HANDLE.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
           MOVE SPACES TO MSGSTR
           STRING 'CSBCTXAL failed' DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
      END-IF.

*********************************
* INTITIALIZE THE CLIENT-LIBRARY *
*********************************

      CALL 'CTBINIT' USING CSL-CTX-HANDLE
                           CSL-RC
                           CS-VERSION-50.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
           MOVE SPACES TO MSGSTR
```

```
           STRING 'CTBINIT failed' DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
       END-IF.

       PERFORM PROCESS-TRANSACTION.

       PERFORM QUIT-CLIENT-LIBRARY.

       GOBACK.



      *=========================================================
       *==                                                     ==
       *== Subroutine to get system date/time                 ==
       *==                                                     ==
       *=========================================================
        GET-SYSTEM-TIME.
       *-----------------

           EXEC CICS ASKTIME
                     ABSTIME(UTIME)
           END-EXEC.

           EXEC CICS FORMATTIME
                     ABSTIME(UTIME)
                     DATESEP('/')
                     MMDDYY(TMP-DATE)
                     TIME(TMP-TIME)
                     TIMESEP
           END-EXEC.

       *=========================================================
       *==                                                     ==
       *== Subroutine to display SYT5 initial screen          ==
       *==                                                     ==
       *=========================================================
        DISPLAY-INITIAL-SCREEN.
       *-----------------------

           MOVE TMP-DATE   TO SDATEO.
           MOVE TMP-TIME   TO STIMEO.
           MOVE 'SYCTSAP5' TO PROGNMO.

           MOVE PAGE-CNT   TO SPAGEO.
```

```
      MOVE MSG-TEXT-1 TO MSG1O.
      MOVE MSG-TEXT-2 TO MSG2O.

      EXEC CICS SEND MAP('A5PANEL')
                     MAPSET('SYCTBA5')
                     CURSOR
                     FRSET
                     ERASE
                     FREEKB
      END-EXEC.

*=========================================================
*==                                                     ==
*== Subroutine to get input data                        ==
*==                                                     ==
*=========================================================
 GET-INPUT-DATA.
*---------------

      EXEC CICS RECEIVE MAP('A5PANEL')
                     MAPSET('SYCTBA5')
                     ASIS
      END-EXEC.

      IF SERVERL = ZERO
        THEN
          IF PF-SERVER = SPACES
            THEN
              MOVE 'Please Enter Server Name' TO MSG-TEXT-1
              MOVE -1                         TO SERVERL
              MOVE C-Y                        TO ENTER-DATA-SW
          END-IF
        ELSE
          MOVE SERVERI   TO PF-SERVER
          MOVE SERVERL   TO PF-SERVER-SIZE
      END-IF.

      IF USERL = ZERO
        THEN
          IF PF-USER = SPACES
            THEN
              MOVE 'Please Enter User-ID' TO MSG-TEXT-1
              MOVE -1                     TO USERL
              MOVE C-Y                    TO ENTER-DATA-SW
          END-IF
        ELSE
```

```
           MOVE USERI   TO PF-USER
           MOVE USERL   TO PF-USER-SIZE
           MOVE PF-USER TO USERO
       END-IF.

       IF PSWDL NOT EQUAL ZERO
         THEN
           MOVE PSWDI TO PF-PWD
           MOVE PSWDL TO PF-PWD-SIZE
       END-IF.

       IF TRANL NOT EQUAL ZERO
         THEN
           MOVE TRANI TO PF-TRAN
           MOVE TRANL TO PF-TRAN-SIZE
       END-IF.

       IF NETDRVL NOT EQUAL ZERO
         THEN
           MOVE NETDRVI TO PF-NETDRV
           MOVE NETDRVL TO PF-DRV-SIZE
       END-IF.

       IF ENTER-DATA-SW = C-Y
         THEN
           MOVE C-N TO ENTER-DATA-SW
           PERFORM DISPLAY-INITIAL-SCREEN
           PERFORM GET-INPUT-DATA
       END-IF.

  *=========================================================
  *==                                                     ==
  *== Subroutine to process input data                    ==
  *==                                                     ==
  *=========================================================
   PROCESS-TRANSACTION.

  ********************************
  * ALLOCATE A CONNECTION HANDLE. *
  ********************************

       MOVE ZERO TO CSL-CON-HANDLE.

       CALL 'CTBCONAL' USING CSL-CTX-HANDLE
                             CSL-RC
                             CSL-CON-HANDLE.
```

```
      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBCONAL failed' DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
      END-IF.

 ******************
 * SET THE USER ID *
 ******************

      CALL 'CTBCONPR' USING CSL-CON-HANDLE
                            CSL-RC
                            CS-SET
                            CS-USERNAME
                            PF-USER
                            PF-USER-SIZE
                            CS-FALSE
                            OUTLEN.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBCONPR for user-id failed' DELIMITED BY SIZE
                                                INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
      END-IF.



******************
 * SET THE PASSWORD *
 ******************

      CALL 'CTBCONPR' USING CSL-CON-HANDLE
                            CSL-RC
                            CS-SET
                            CS-PASSWORD
                            PF-PWD
                            PF-PWD-SIZE
                            CS-FALSE
                            OUTLEN.
```

```
      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBCONPR for password failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
      END-IF.

*********************
* SET THE TRAN NAME *
*********************

      IF PF-TRAN-SIZE IS NOT EQUAL TO ZEROES THEN

      CALL 'CTBCONPR' USING CSL-CON-HANDLE
                            CSL-RC
                            CS-SET
                            CS-TRANSACTION-NAME
                            PF-TRAN
                            PF-TRAN-SIZE
                            CS-FALSE
                            OUTLEN

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBCONPR for TRANname failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
        END-IF

      END-IF.

*******************************
* SET THE NET DRIVER PROPERTY *
*******************************

      IF PF-NETDRV = SPACES OR PF-NETDRV = 'LU62'                 X
                            OR PF-NETDRV = 'lu62'
          MOVE CS-LU62 TO NETDRIVER
      ELSE
        IF PF-NETDRV = 'IBMTCPIP' OR PF-NETDRV = 'ibmtcpip'
          MOVE CS-TCPIP TO NETDRIVER
      ELSE
```

```
          IF PF-NETDRV = 'INTERLIN' OR PF-NETDRV = 'interlin'
            MOVE CS-INTERLINK TO NETDRIVER
        ELSE
          IF PF-NETDRV = 'CPIC' OR PF-NETDRV = 'cpic'
            MOVE CS-NCPIC TO NETDRIVER
        END-IF.

        IF PF-DRV-SIZE IS NOT EQUAL TO ZEROES THEN

            CALL 'CTBCONPR' USING CSL-CON-HANDLE
                                  CSL-RC
                                  CS-SET
                                  CS-NET-DRIVER
                                  NETDRIVER
                                  CS-UNUSED
                                  CS-FALSE
                                  OUTLEN

            IF CSL-RC NOT EQUAL CS-SUCCEED
              THEN
                 MOVE SPACES TO MSGSTR
                 STRING 'CTBCONPR for network driver failed'
                        DELIMITED BY SIZE INTO MSGSTR
                 PERFORM PRINT-MSG
                 PERFORM ALL-DONE
              END-IF

        END-IF.


      **********************************
      * SETUP retrieval of All Messages *
      **********************************

        CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                             CSL-RC,
                             CS-UNUSED,
                             CS-INIT,
                             CS-ALLMSG-TYPE,
                             CS-UNUSED,
                             CS-UNUSED.

        IF CSL-RC NOT EQUAL CS-SUCCEED
          THEN
            MOVE SPACES TO MSGSTR
            STRING 'CTBDIAG CS-INIT failed' DELIMITED BY SIZE
```

```
                                                    INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.

*******************************************
* set the upper limit of number of messages *
*******************************************

      MOVE 5 TO PF-MSGLIMIT.

      CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                           CSL-RC,
                           CS-UNUSED,
                           CS-MSGLIMIT,
                           CS-ALLMSG-TYPE,
                           CS-UNUSED,
                           PF-MSGLIMIT.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBDIAG CS-MSGLIMIT failed' DELIMITED BY SIZE
                                                    INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.

***********************************************
* CONNECT TO THE SERVER OR THE IMS/CICS REGION *
***********************************************

      CALL 'CTBCONNE' USING CSL-CON-HANDLE
                            CSL-RC
                            PF-SERVER
                            PF-SERVER-SIZE
                            CS-FALSE.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBCONNE failed' DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
      END-IF
```

```
     IF NO-ERRORS
       THEN
         PERFORM SEND-COMMAND
     END-IF

************************************
* PROCESS THE RESULTS OF THE COMMAND *
************************************

     IF NO-ERRORS
       THEN
         PERFORM RESULTS-PROCESSING UNTIL NO-MORE-RESULTS
         PERFORM CLOSE-CONNECTION
     END-IF.

 PROCESS-TRANSACTION-EXIT.
     EXIT.

*============================================================
*==                                                        ==
*== Subroutine to allocate, send, and process commands     ==
*==                                                        ==
*============================================================
 SEND-COMMAND.

*-------------------------------------------------------------
*  find out what the maximum number of connections is
*-------------------------------------------------------------
     CALL 'CTBCONFI'  USING CSL-CTX-HANDLE,
                            CSL-RC,
                            CS-GET,
                            CS-MAX-CONNECT,
                            CF-MAXCONNECT,
                            CF-FOUR,
                            CS-FALSE,
                            CF-OUTLEN.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBCONFI CS-GET failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.
```

```
*-------------------------------------------------------------
*  allocate a command handle
*-------------------------------------------------------------

     CALL 'CTBCMDAL' USING CSL-CON-HANDLE,
                           CSL-RC,
                           CSL-CMD-HANDLE.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBCMDAL failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

*-------------------------------------------------------------
*    prepare the language request
*-------------------------------------------------------------

     MOVE CF-LANG2-SIZE TO PF-STRLEN.

     CALL 'CTBCOMMA' USING CSL-CMD-HANDLE,
                           CSL-RC,
                           CS-LANG-CMD,
                           CF-LANG2,
                           PF-STRLEN,
                           CS-UNUSED.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBCOMMA CS-LANG-CMD failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

*-------------------------------------------------------------
*    send the language request
*-------------------------------------------------------------
     CALL 'CTBSEND' USING CSL-CMD-HANDLE,
                          CSL-RC.

     IF CSL-RC NOT EQUAL CS-SUCCEED
```

```
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBSEND failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

 SEND-COMMAND-EXIT.
     EXIT.

*=========================================================
*==                                                     ==
*== Subroutine to process result                        ==
*==                                                     ==
*=========================================================
 RESULTS-PROCESSING.

**************************
* SET UP THE RESULTS DATA *
**************************

     CALL 'CTBRESUL' USING CSL-CMD-HANDLE
                           CSL-RC
                           RESTYPE.

************************************************
* DETERMINE THE OUTCOME OF THE COMMAND EXECUTION *
************************************************

     EVALUATE CSL-RC

         WHEN CS-SUCCEED

*****************************************************************
* DETERMINE THE TYPE OF RESULT RETURNED BY THE CURRENT REQUEST *
*****************************************************************

             EVALUATE RESTYPE

***********************
* PROCESS ROW RESULTS *
***********************

               WHEN CS-ROW-RESULT
                 PERFORM RESULT-ROW-PROCESSING
```

```
                MOVE 'Y' TO SW-FETCH
                PERFORM FETCH-ROW-PROCESSING UNTIL NO-MORE-ROWS

 ************************************************************
 * PROCESS PARAMETER RESULTS - THERE SHOULD BE NO PARAMETERS *
 * TO PROCESS                                                *
 ************************************************************

              WHEN CS-PARAM-RESULT
                MOVE 'Y' TO SW-FETCH

 ************************************************************
 * PROCESS STATUS RESULTS - THE STORED PROCEDURE STATUS RESULT *
 * WILL NOT BE PROCESSED IN THIS EXAMPLE                      *
 ************************************************************

              WHEN CS-STATUS-RESULT
                MOVE 'Y' TO SW-FETCH



***********************************************************
 * PRINT AN ERROR MESSAGE IF THE SERVER ENCOUNTERED AN ERROR *
 * WHILE EXECUTING THE REQUEST                               *
 ***********************************************************

              WHEN CS-CMD-FAIL
                STRING
                   'CTBRESUL returned CS-CMD-FAIL restype'
                   DELIMITED BY SIZE INTO MSGSTR
                PERFORM PRINT-MSG

 ************************************************************
 * PRINT A MESSAGE FOR SUCCESSFUL COMMANDS THAT RETURNED NO DATA *
 * (OPTIONAL)                                                *
 ************************************************************
              WHEN CS-CMD-SUCCEED
                STRING
                   'CTBRESUL returned CS-CMD-SUCCEED restype'
                   DELIMITED BY SIZE INTO MSGSTR

 ************************************************************
 * PRINT A MESSAGE FOR REQUESTS THAT HAVE BEEN PROCESSED *
 * SUCCESSFULLY (OPTIONAL)                                   *
 ************************************************************
```

```
                WHEN CS-CMD-DONE
                  STRING 'CTBRESUL returned CS-CMD-DONE restype'
                              DELIMITED BY SIZE INTO MSGSTR

                WHEN OTHER
                   STRING 'CTBRESUL returned UNKNOWN restype'
                              DELIMITED BY SIZE INTO MSGSTR
                   PERFORM PRINT-MSG
                   MOVE 'N' TO SW-RESULTS

              END-EVALUATE

*******************************************************
* PRINT AN ERROR MESSAGE IF THE CTBRESULTS CALL FAILED *
*******************************************************

         WHEN CS-FAIL
           MOVE 'N' TO SW-RESULTS
           STRING 'CTBRESUL returned CS-FAIL ret-code'
                     DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG

*************************************************************
* DROP OUT OF THE RESULTS LOOP IF NO MORE RESULT SETS ARE   *
* AVAILABLE FOR PROCESSING OR IF THE RESULTS WERE CANCELLED *
*************************************************************

         WHEN CS-END-RESULTS
           MOVE 'N' TO SW-RESULTS

         WHEN CS-CANCELLED
           MOVE 'N' TO SW-RESULTS

         WHEN OTHER
           MOVE 'N' TO SW-RESULTS
           STRING 'CTBRESUL returned UNKNOWN ret-code'
                     DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG

     END-EVALUATE.

     MOVE 0 TO RESTYPE.

 RESULTS-PROCESSING-EXIT.
     EXIT.
```

```
*=========================================================
*==                                                     ==
*== Subroutine to process result rows                   ==
*==                                                     ==
*=========================================================
 RESULT-ROW-PROCESSING.

     CALL 'CTBRESIN' USING CSL-CMD-HANDLE,
                           CSL-RC,
                           CS-NUMDATA,
                           RF-NUMDATA,
                           RF-NUMDATA-SIZE,
                           CF-COL-LEN.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBRESINFO failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

     COMPUTE FF-ROW-NUM = FF-ROW-NUM + 1.

*********************************
* display number of connections *
*********************************

     MOVE CF-MAXCONNECT   TO OR2-MAXCONNECT.
     MOVE OUTPUT-ROW-STR2 TO RSLTNO(FF-ROW-NUM).
     COMPUTE FF-ROW-NUM = FF-ROW-NUM + 2.

*********************************
* display the number of columns *
*********************************

     MOVE RF-NUMDATA      TO OR4-NUMDATA.
     MOVE OUTPUT-ROW-STR4 TO RSLTNO(FF-ROW-NUM).

     IF RF-NUMDATA NOT EQUAL 3
       THEN
         STRING 'CTBRESINFO returned wrong # of parms' DELIMITED
                                       BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
```

```
     END-IF.

     COMPUTE FF-ROW-NUM = FF-ROW-NUM + 2.

**-------------------------------------------------------------
**   Setup column headings
**-------------------------------------------------------------

     MOVE '  PLANEID               Milage                    Servic
-    'e Date'   TO RSLTNO(FF-ROW-NUM).
     COMPUTE FF-ROW-NUM = FF-ROW-NUM + 1.
     MOVE '============ ============================== ==========
-    '===============' TO RSLTNO(FF-ROW-NUM).
     PERFORM BIND-COLUMNS
        VARYING I FROM 1 BY 1
           UNTIL I IS GREATER THAN RF-NUMDATA.

 RESULT-ROW-PROCESSING-EXIT.
     EXIT.

*=========================================================
*==                                                     ==
*== Subroutine to bind each data                        ==
*==                                                     ==
*=========================================================
 BIND-COLUMNS.

      CALL 'CTBDESCR' USING CSL-CMD-HANDLE,
                            CSL-RC,
                            I,
                            DATAFMT.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBDESCR failed'
                 DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.

**-------------------------------------------------------------
**   We need to bind the data to program variables.
**   We don't care about the indicator variable
**   so we'll pass NULL for that parameter in OC-BIND().
**-------------------------------------------------------------
```

```
******************
* ROWs per FETCH *
******************
       MOVE 1 TO DF-COUNT

       EVALUATE DF-DATATYPE

         WHEN CS-DECIMAL-TYPE
           MOVE LENGTH OF CF-COL-MILAGE         TO DF-MAXLENGTH
           CALL 'CTBBIND' USING CSL-CMD-HANDLE,
                                CSL-RC,
                                I,
                                DATAFMT,
                                CF-COL-MILAGE-NUMBER
                                CF-COL-LEN,
                                CS-PARAM-NOTNULL,
                                CF-COL-INDICATOR,
                                CS-PARAM-NULL

           IF CSL-RC NOT EQUAL CS-SUCCEED
             THEN
               MOVE SPACES TO MSGSTR
               STRING 'CTBBIND CS-DECIMAL-TYPE Filed' DELIMITED
                                        BY SIZE INTO MSGSTR
               PERFORM PRINT-MSG
               PERFORM ALL-DONE
           END-IF

         WHEN CS-VARCHAR-TYPE

           MOVE LENGTH OF CF-COL-FIRSTNME-TXT TO DF-MAXLENGTH

           CALL 'CTBBIND' USING CSL-CMD-HANDLE,
                                CSL-RC,
                                I,
                                DATAFMT,
                                CF-COL-FIRSTNME,
                                CF-COL-LEN,
                                CS-PARAM-NOTNULL,
                                CF-COL-INDICATOR,
                                CS-PARAM-NULL

           IF CSL-RC NOT EQUAL CS-SUCCEED
             THEN
               MOVE SPACES TO MSGSTR
```

```
               STRING 'CTBBIND CS-VARCHAR-TYPE failed' DELIMITED
                                       BY SIZE INTO MSGSTR
               PERFORM PRINT-MSG
               PERFORM ALL-DONE
             END-IF
           WHEN CS-DATETIME-TYPE

             MOVE LENGTH OF CF-COL-SERVICEDATE-BOUND
                                TO DF-MAXLENGTH
             MOVE LENGTH OF CF-COL-SERVICEDATE-BOUND TO CF-COL-LEN

             CALL 'CTBBIND' USING CSL-CMD-HANDLE,
                                CSL-RC,
                                I,
                                DATAFMT,
                                CF-COL-SERVICEDATE-BOUND,
                                CF-COL-LEN,
                                CS-PARAM-NOTNULL,
                                CF-COL-INDICATOR,
                                CS-PARAM-NULL

             IF CSL-RC NOT EQUAL CS-SUCCEED
               THEN
                 MOVE SPACES TO MSGSTR
                 STRING 'CTBBIND CS-DATETIME-TYPE failed' DELIMITED
                                       BY SIZE INTO MSGSTR
                 PERFORM PRINT-MSG
                 PERFORM ALL-DONE
             END-IF.

 BIND-COLUMNS-EXIT.
     EXIT.

*=========================================================
*==                                                     ==
*== Subroutine to fetch row processing                  ==
*==                                                     ==
*=========================================================
 FETCH-ROW-PROCESSING.

     CALL 'CTBFETCH' USING CSL-CMD-HANDLE,
                           CSL-RC,
                           CS-UNUSED,
                           CS-UNUSED,
                           CS-UNUSED,
                           FF-ROWS-READ.
```

```
       EVALUATE CSL-RC

           WHEN CS-SUCCEED
                MOVE 'Y'              TO SW-FETCH
                MOVE CS-VARCHAR-TYPE TO DF-DATATYPE
                MOVE LENGTH OF CF-COL-FIRSTNME-TXT
                                      TO DF-MAXLENGTH
                MOVE CS-CHAR-TYPE    TO DF2-DATATYPE
                MOVE LENGTH OF CF-COL-FIRSTNME-CHAR
                                      TO DF2-MAXLENGTH

                CALL 'CSBCONVE' USING CSL-CTX-HANDLE,
                                      CSL-RC,
                                      DATAFMT,
                                      CF-COL-FIRSTNME,
                                      DATAFMT2,
                                      CF-COL-FIRSTNME-CHAR,
                                      CF-COL-LEN

                IF CSL-RC NOT EQUAL CS-SUCCEED
                  THEN
                     MOVE SPACES TO MSGSTR
                     STRING 'CSBCONVERT CS-VARCHAR-TYPE failed'
                                 DELIMITED BY SIZE INTO MSGSTR
                     PERFORM PRINT-MSG
                     PERFORM ALL-DONE
                END-IF

                MOVE CS-DECIMAL-TYPE TO DF-DATATYPE
      *         MOVE LENGTH OF CF-COL-MILAGE-NUMBER
      *
                MOVE 35              TO DF-MAXLENGTH
                MOVE CS-PACKED370-TYPE    TO DF2-DATATYPE
                MOVE LENGTH OF CF-COL-MILAGE-DECFORM
                                      TO DF2-MAXLENGTH

                CALL 'CSBCONVE' USING CSL-CTX-HANDLE,
                                      CSL-RC,
                                      DATAFMT,
                                      CF-COL-MILAGE-NUMBER,
                                      DATAFMT2,
                                      CF-COL-MILAGE-DECFORM,
                                      CF-COL-LEN

                IF CSL-RC NOT EQUAL CS-SUCCEED
```

```
                  THEN
                     MOVE SPACES TO MSGSTR
                     STRING 'CSBCONVERT from CS_DECIMAL to CS-PACKED
-                    '370-TYPE FAILED'
                                 DELIMITED BY SIZE INTO MSGSTR
                     PERFORM PRINT-MSG
                     PERFORM ALL-DONE
                END-IF
                MOVE LENGTH OF CF-COL-SERVICEDATE-BOUND
                                      TO DF-MAXLENGTH
                MOVE CS-DATETIME-TYPE TO DF-DATATYPE
                MOVE CS-CHAR-TYPE     TO DF2-DATATYPE
                MOVE LENGTH OF CF-COL-SERVICEDATE-CHAR
                                      TO DF2-MAXLENGTH

                CALL 'CSBCONVE' USING CSL-CTX-HANDLE,
                                      CSL-RC,
                                      DATAFMT,
                                      CF-COL-SERVICEDATE-BOUND,
                                      DATAFMT2,
                                      CF-COL-SERVICEDATE-CHAR,
                                      CF-COL-LEN

                IF CSL-RC NOT EQUAL CS-SUCCEED
                  THEN
                     MOVE SPACES TO MSGSTR
                     STRING 'CSBCONVERT from DATETIME   to CS-CHAR f
-                    'ailed'
                                 DELIMITED BY SIZE INTO MSGSTR
                     PERFORM PRINT-MSG
                     PERFORM ALL-DONE
                END-IF

                COMPUTE FF-ROW-NUM = FF-ROW-NUM + 1

           ************************************
           * save ROW RESULTS for later display *
           ************************************

                MOVE CF-COL-FIRSTNME-CHAR TO
                     OR-COL-FIRSTNME-CHAR
                MOVE CF-COL-MILAGE-DECFORM TO
                     OR-COL-MILAGE
                MOVE CF-COL-SERVICEDATE-CHAR TO
                     OR-COL-SERVICEDATE
                IF FF-ROW-NUM > MAX-SCREEN-ROWS
```

```
                THEN
                  STRING 'Please press return for more data.'
                         DELIMITED BY SIZE INTO MSG1O
                  PERFORM DISP-DATA
                  PERFORM CLEAR-SCREEN-DATA
                          VARYING I2 FROM 1 BY 1
                          UNTIL I2 > MAX-SCREEN-ROWS
                  MOVE 1      TO FF-ROW-NUM
**------------------------------------------------------------
**   Setup column headings
**------------------------------------------------------------
                  MOVE '  PLANEID               Milage
-                 '        Service Date  '
                  TO RSLTNO(FF-ROW-NUM)
                  COMPUTE FF-ROW-NUM = FF-ROW-NUM + 1
                  MOVE '============ ==============================
-                 '== ========================'
                  TO  RSLTNO(FF-ROW-NUM)
                  COMPUTE FF-ROW-NUM = FF-ROW-NUM + 1
              END-IF

              MOVE OUTPUT-ROW-STR TO RSLTNO(FF-ROW-NUM)

              MOVE SPACES         TO CF-COL-FIRSTNME-TXT

         WHEN CS-END-DATA
              MOVE SPACES TO MSG1O
              MOVE 'N'    TO SW-FETCH
              STRING 'All rows processing completed!'
                     DELIMITED BY SIZE INTO MSG1O
              PERFORM DISP-DATA

         WHEN CS-FAIL
              MOVE 'N'    TO SW-FETCH
              MOVE SPACES TO MSGSTR
              STRING 'CTBFETCH returned CS-FAIL ret-code'
                     DELIMITED BY SIZE INTO MSGSTR
              PERFORM PRINT-MSG

         WHEN CS-ROW-FAIL
              MOVE 'N'    TO SW-FETCH
              MOVE SPACES TO MSGSTR
              STRING 'CTBFETCH returned CS-ROW-FAIL ret-code'
                              DELIMITED BY SIZE INTO MSGSTR
              PERFORM PRINT-MSG
```

```
           WHEN CS-CANCELLED
                MOVE 'N'          TO SW-FETCH
                MOVE MF-CANCELED TO MSG1O
                PERFORM PRINT-MSG

           WHEN OTHER
                MOVE 'N'    TO SW-FETCH
                MOVE SPACES TO MSGSTR
                STRING 'CTBFETCH returned UNKNOWN ret-code'
                                DELIMITED BY SIZE INTO MSGSTR
                PERFORM PRINT-MSG

       END-EVALUATE.

 FETCH-ROW-PROCESSING-EXIT.
       EXIT.

*=========================================================
*==                                                     ==
*== Subroutine to tell CICS to send output messages     ==
*==                                                     ==
*=========================================================
 DISP-DATA.

***********************************************************
* PRINT ALL THE RETURNED ROWS FROM THE STORED PROCEDURE *
***********************************************************

       MOVE TMP-DATE   TO SDATEO.
       MOVE TMP-TIME   TO STIMEO.
       MOVE 'SYCTSAP5' TO PROGNMO.
       MOVE PAGE-CNT   TO SPAGEO.

       MOVE DFHBMPRO   TO SERVERA.
       MOVE PF-SERVER  TO SERVERO.

       MOVE DFHBMPRO   TO USERA.
       MOVE PF-USER    TO USERO.

       MOVE DFHBMPRO   TO NETDRVA.
       MOVE PF-NETDRV  TO NETDRVO.

       MOVE DFHBMDAR   TO PSWDA.
       MOVE PF-PWD    TO PSWDO.
       MOVE MSG-TEXT-2 TO MSG2O.
```

```
*********************
* DISPLAY THE DATA *
*********************

     EXEC CICS SEND MAP('A5PANEL')
                    MAPSET('SYCTBA5')
                    CURSOR
                    FRSET
                    ERASE
                    FREEKB
     END-EXEC.

     EXEC CICS RECEIVE INTO(QF-ANSWER)
                       LENGTH(QF-LEN)
                       MAXLENGTH(QF-MAXLEN)
                       RESP(CICS-RESPONSE)
     END-EXEC.

 DISP-DATA-EXIT.
     EXIT.

*============================================================
*==                                                        ==
*== Subroutine to print output messages.                   ==
*==                                                        ==
*============================================================
 PRINT-MSG.

     MOVE LOW-VALUES TO A5PANELO.
     MOVE CSL-RC  TO SAMP-RC.
     MOVE RESTYPE TO REST-TYPE.

     IF DIAG-MSGS-INITIALIZED
       THEN
         PERFORM GET-DIAG-MESSAGES
     END-IF.

************************
* DISPLAY THE MESSAGE *
************************

     MOVE DISP-MSG TO MSG1O.

     IF NO-ERRORS
       THEN
         PERFORM DISP-DATA.
```

```
           MOVE C-Y    TO NO-ERRORS-SW.
           MOVE SPACES TO MSGSTR.
           MOVE SPACES TO MSG1O.
           MOVE ZERO   TO SAMP-RC.
           MOVE ZERO   TO REST-TYPE.

       PRINT-MSG-EXIT.
           EXIT.

      *============================================================
      *==                                                        ==
      *== Subroutine to drop and to deallocate all handlers,     ==
      *== to close server connection and exit client library     ==
      *==                                                        ==
      *============================================================
       ALL-DONE.

           PERFORM CLOSE-CONNECTION.
           PERFORM QUIT-CLIENT-LIBRARY.
           STOP RUN.

       ALL-DONE-EXIT.
           EXIT.

      *================================================================
      *==                                                            ==
      *== Subroutine to perform drop command handler, close          ==
      *== server connection, and deallocate Connection Handler.      ==
      *==                                                            ==
      *================================================================
       CLOSE-CONNECTION.



      ***************************
       * DROP THE COMMAND HANDLE *
       ***************************

           CALL 'CTBCMDDR' USING CSL-CMD-HANDLE
                                 CSL-RC.

           IF CSL-RC = CS-FAIL
             THEN
               MOVE SPACES TO MSGSTR
               STRING 'CTBCMDDR failed' DELIMITED BY
```

```
                    SIZE INTO MSGSTR
               PERFORM PRINT-MSG
          END-IF.

     ******************************
     * CLOSE THE SERVER CONNECTION *
     ******************************

          CALL 'CTBCLOSE' USING CSL-CON-HANDLE
                                CSL-RC
                                CS-UNUSED.

          IF CSL-RC = CS-FAIL
            THEN
              MOVE SPACES TO MSGSTR
              STRING 'CTBCLOSE failed' DELIMITED BY
                      SIZE INTO MSGSTR
              PERFORM PRINT-MSG
          END-IF.

     ***********************************
     * DE-ALLOCATE THE CONNECTION HANDLE *
     ***********************************

          CALL 'CTBCONDR' USING CSL-CON-HANDLE
                                CSL-RC.

          IF CSL-RC = CS-FAIL
            THEN
              MOVE SPACES TO MSGSTR
              STRING 'CTBCONDR failed' DELIMITED BY
                      SIZE INTO MSGSTR
              PERFORM PRINT-MSG
          END-IF.

      CLOSE-CONNECTION-EXIT.
          EXIT.



    *================================================
     *==                                            ==
     *== Subroutine to perform exit client library and ==
     *== deallocate context structure.              ==
     *==                                            ==
     *================================================
```

```
 QUIT-CLIENT-LIBRARY.

**************************
* EXIT THE CLIENT LIBRARY *
**************************

     CALL 'CTBEXIT' USING CSL-CTX-HANDLE
                          CSL-RC
                          CS-UNUSED.
     IF CSL-RC = CS-FAIL
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBEXIT failed' DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
     END-IF.

***********************************
* DE-ALLOCATE THE CONTEXT STRUCTURE *
***********************************

     CALL 'CSBCTXDR' USING CSL-CTX-HANDLE
                           CSL-RC.

     IF CSL-RC = CS-FAIL
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CSBCTXDR failed' DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
     END-IF.

 QUIT-CLIENT-LIBRARY-EXIT.
     EXIT.

*=====================================================
*==                                                 ==
*== Subroutine to retrieve any diagnostic messages ==
*==                                                 ==
*=====================================================
 GET-DIAG-MESSAGES.

***********************************
* Disable calls to this subroutine *
***********************************

     MOVE 'N' TO SW-DIAG.
```

```
      ******************************
      * First, get client messages *
      ******************************

           CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                                CSL-RC,
                                CS-UNUSED,
                                CS-STATUS,
                                CS-CLIENTMSG-TYPE,
                                CS-UNUSED,
                                DG-NUM-OF-MSGS.

           IF CSL-RC NOT EQUAL CS-SUCCEED
             THEN
               MOVE SPACES TO MSGSTR
               STRING 'CTBDIAG CS-STATUS CS-CLIENTMSG-TYP fail'
                               DELIMITED BY SIZE INTO MSGSTR
               PERFORM PRINT-MSG
               PERFORM ALL-DONE
             ELSE
               IF DG-NUM-OF-MSGS > 0
                 THEN
                   PERFORM RETRIEVE-CLIENT-MSGS
                       VARYING I FROM 1 BY 1
                           UNTIL I IS GREATER THAN DG-NUM-OF-MSGS
               END-IF
           END-IF.



     ******************************
      * Then, get server messages *
      ******************************

           CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                                CSL-RC,
                                CS-UNUSED,
                                CS-STATUS,
                                CS-SERVERMSG-TYPE,
                                CS-UNUSED,
                                DG-NUM-OF-MSGS.

           IF CSL-RC NOT EQUAL CS-SUCCEED
             THEN
               STRING 'CTBDIAG CS-STATUS CS-SERVERMSG-TYP fail'
                               DELIMITED BY SIZE INTO MSGSTR
```

```
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
        ELSE
          IF DG-NUM-OF-MSGS > 0
            THEN
              PERFORM RETRIEVE-SERVER-MSGS
                  VARYING I FROM 1 BY 1
                      UNTIL I IS GREATER THAN DG-NUM-OF-MSGS
          END-IF
      END-IF.

 GET-DIAG-MESSAGES-EXIT.
      EXIT.

*============================================================
*==                                                        ==
*== Subroutine to retrieve diagnostic messages from client ==
*==                                                        ==
*============================================================
 RETRIEVE-CLIENT-MSGS.

      MOVE 1 TO I1.

      CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                           CSL-RC,
                           CS-UNUSED,
                           CS-GET,
                           CS-CLIENTMSG-TYPE,
                           DG-MSGNO,
                           CLIENT-MSG.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBDIAG CS-GET CS-CLIENTMSG-TYPE failed'
                         DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.

************************
* display message text *
************************

      MOVE DISP-CLIENT-MSG-HDR TO RSLTNO( I1 ).
      MOVE 3 TO I1.
```

```
MOVE CM-SEVERITY       TO CM-SEVERITY-DATA.
MOVE CM-STATUS         TO CM-STATUS-DATA.
MOVE DISP-CLIENT-MSG-1 TO RSLTNO( I1 ).
COMPUTE I1 EQUAL I1 + 1

MOVE CM-MSGNO          TO CM-OC-MSGNO-DATA.
MOVE DISP-CLIENT-MSG-2 TO RSLTNO( I1 ).
COMPUTE I1 EQUAL I1 + 1

IF CM-MSGNO NOT EQUAL 0
  THEN
    MOVE SPACES            TO CM-OC-MSG-DATA
    MOVE CM-TEXT           TO CM-OC-MSG-DATA
    MOVE CM-TEXT           TO DISP-CLIENT-MSG-3A
    MOVE DISP-CLIENT-MSG-3 TO RSLTNO( I1 )
    COMPUTE I1 EQUAL I1 + 1
    IF CM-TEXT-LEN > 66
      THEN
        MOVE CM-OC-MSG-DATA-2   TO CM-OC-MSG-DATA-X
        MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
        COMPUTE I1 EQUAL I1 + 1
        IF CM-TEXT-LEN > 132
          THEN
            MOVE SPACES             TO CM-OC-MSG-DATA-X
            MOVE CM-OC-MSG-DATA-3   TO CM-OC-MSG-DATA-X
            MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
            COMPUTE I1 EQUAL I1 + 1
            IF CM-TEXT-LEN > 198
              THEN
                MOVE SPACES             TO CM-OC-MSG-DATA-X
                MOVE CM-OC-MSG-DATA-4   TO CM-OC-MSG-DATA-X
                MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
                COMPUTE I1 EQUAL I1 + 1
            END-IF
        END-IF
      END-IF
  ELSE
    MOVE DISP-EMPTY-CLIENT-MSG-3 TO RSLTNO( I1 )
    COMPUTE I1 EQUAL I1 + 1
END-IF.

MOVE CM-OS-MSGNO       TO CM-OS-MSGNO-DATA.
MOVE DISP-CLIENT-MSG-4 TO RSLTNO( I1 ).
COMPUTE I1 EQUAL I1 + 1
```

```
      IF CM-OS-MSGNO NOT EQUAL 0
        THEN
          MOVE SPACES             TO CM-OS-MSG-DATA
          MOVE CM-OS-MSGTXT       TO CM-OS-MSG-DATA
          MOVE SPACES             TO DISP-CLIENT-MSG-5A
          MOVE CM-OS-MSGTXT       TO DISP-CLIENT-MSG-5A
          MOVE DISP-CLIENT-MSG-5 TO RSLTNO( I1 )
          COMPUTE I1 EQUAL I1 + 1
          IF CM-OS-MSGTEXT-LEN > 66
            THEN
              MOVE SPACES             TO CM-OC-MSG-DATA-X
              MOVE CM-OS-MSG-DATA-2   TO CM-OC-MSG-DATA-X
              MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
              COMPUTE I1 EQUAL I1 + 1
              IF CM-OS-MSGTEXT-LEN > 132
                THEN
                  MOVE SPACES             TO CM-OC-MSG-DATA-X
                  MOVE CM-OS-MSG-DATA-3   TO CM-OC-MSG-DATA-X
                  MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
                  COMPUTE I1 EQUAL I1 + 1
                  IF CM-OS-MSGTEXT-LEN > 198
                    THEN
                      MOVE SPACES             TO CM-OC-MSG-DATA-X
                      MOVE CM-OS-MSG-DATA-4   TO CM-OC-MSG-DATA-X
                      MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
                      COMPUTE I1 EQUAL I1 + 1
                  END-IF
              END-IF
          END-IF
        ELSE
          MOVE DISP-EMPTY-CLIENT-MSG-5 TO RSLTNO( I1 )
          COMPUTE I1 EQUAL I1 + 1
      END-IF.

 RETRIEVE-CLIENT-MSGS-EXIT.
      EXIT.

*============================================================
*==                                                        ==
*== Subroutine to retrieve diagnostic messages from server ==
*==                                                        ==
*============================================================
 RETRIEVE-SERVER-MSGS.

       CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                            CSL-RC,
```

```
                            CS-UNUSED,
                            CS-GET,
                            CS-SERVERMSG-TYPE,
                            DG-MSGNO,
                            SERVER-MSG.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBDIAG CS-GET CS-SERVERMSG-TYPE failed'
                         DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

************************
* display message text *
************************

     MOVE SM-MSGNO   TO SM-MSG-NO-DATA.
     MOVE SM-SEV     TO SM-SEVERITY-DATA.
     MOVE SM-STATE   TO SM-STATE-DATA.

     MOVE SM-LINE    TO SM-LINE-NO-DATA.
     MOVE SM-STATUS  TO SM-STATUS-DATA.

     MOVE SPACES     TO SM-SVRNAME-DATA.
     MOVE SM-SVRNAME TO SM-SVRNAME-DATA.

     MOVE SPACES     TO SM-PROC-ID-DATA.
     MOVE SM-PROC    TO SM-PROC-ID-DATA.

     MOVE SPACES     TO SM-MSG-DATA.
     MOVE SM-TEXT    TO SM-MSG-DATA.

     MOVE SPACES     TO DISP-SERVER-MSG-5A.
     MOVE SM-TEXT    TO DISP-SERVER-MSG-5A.


     MOVE DISP-SERVER-MSG-HDR TO RSLTNO (1).
     MOVE DISP-SERVER-MSG-1   TO RSLTNO (3).
     MOVE DISP-SERVER-MSG-2   TO RSLTNO (4).
     MOVE DISP-SERVER-MSG-3   TO RSLTNO (5).
     MOVE DISP-SERVER-MSG-4   TO RSLTNO (6).

     MOVE DISP-SERVER-MSG-5   TO RSLTNO (7).
```

```
      IF SM-TEXT-LEN > 66
        THEN
          MOVE SPACES              TO SM-MSG-DATA-X
          MOVE SM-MSG-DATA-2       TO SM-MSG-DATA-X
          MOVE DISP-SERVER-MSG-5X TO RSLTNO(8)
          IF SM-TEXT-LEN > 132
            THEN
              MOVE SPACES              TO SM-MSG-DATA-X
              MOVE SM-MSG-DATA-3       TO SM-MSG-DATA-X
              MOVE DISP-SERVER-MSG-5X TO RSLTNO(9)
              IF SM-TEXT-LEN > 198
                THEN
                  MOVE SPACES              TO SM-MSG-DATA-X
                  MOVE SM-MSG-DATA-4       TO SM-MSG-DATA-X
                  MOVE DISP-SERVER-MSG-5X TO RSLTNO(10)
              END-IF
          END-IF
      END-IF.

  RETRIEVE-SERVER-MSGS-EXIT.
      EXIT.



*============================================================
 *==                                                        ==
 *== Subroutine to clear the output screen                  ==
 *==                                                        ==
 *============================================================
  CLEAR-SCREEN-DATA.

      MOVE SPACES TO RSLTNO( I2 ).

  CLEAR-SCREEN-DATA-EXIT.

      EXIT.



*============================================================
 *==                                                        ==
 *== Subroutine to handle MAPFAIL condition                 ==
 *==                                                        ==
 *============================================================
  NO-INPUT.
 *---------
```

```
     MOVE 'Please Enter Input Fields' TO MSG-TEXT-1.

     GO TO GET-INPUT-AGAIN.

*=========================================================
*==                                                     ==
*== Subroutine to handle AID condition                  ==
*==                                                     ==
*=========================================================
 GETOUT.
*--------

     EXEC CICS RETURN END-EXEC.

     STOP RUN.

*=========================================================
*==                                                     ==
*== Subroutine to handle ERROR condition                ==
*==                                                     ==
*=========================================================
 ERRORS.
*--------

     EXEC CICS DUMP DUMPCODE('ERRS') END-EXEC.

     STOP RUN.
```

SYCTSAT5 - sample language request

The purpose of this sample program is to demonstrate:

- Implicit conversions of CS-VARCHAR and CS-DECIMAL datatypes to CS-CHAR
- How to transform a CS-DATETIME datatype to a DB2 ISO DATE and TIME format

This sample program retrieves information from the table, SYBASE.NEWTABLE on the target server.

```
*@(#) syctsat5.cobol 1.2 4/26/96        */
```

```
*****************************************************************
*
* Confidential property of Sybase, Inc.
* (c) Copyright Sybase, Inc. 1985 TO 1997.
* All rights reserved.
*
*****************************************************************

******* SYCTSAT5 - Client Language Request APPL - COBOL - CICS **
**
**  CICS TRANID:  SYT5
**
**  PROGRAM:  SYCTSAT5
**
**  PURPOSE:  Demonstrates Open Client for CICS CALLs.
**
**  FUNCTION: Illustrates how to send a language request to
**            a SQL Server.
**
**            Illustrates the implicit conversion of
**            DECIMAL to CHAR  data type
**
**              The request  sent to  SQL Server
**              executes the SQL statement:
**
**               SELECT  PLANEID, MILAGE,
**               CONVERT (CHAR(10),SERVICEDATE,102)+" " +
**               CONVERT (CHAR(8),SERVICEDATE,108)
**                       FROM  SYBASE.NEWTABLE
**
**  PREREQS:  Before running SYCTSAT5, make sure that the server
**            you wish to access has an entry in the Connection
**            Router Table for that Server and the MCG(s) that
**            you wish to use.
**
**  INPUT:    On the input screen, make sure to enter the Server
**            name, user id, and password for the target server.
**            TRAN NAME is not used for LAN servers.
**
**
**  Open Client CALLs used in this sample:
**
**    CSBCTXALLOC   allocate a context
**    CSBCTXDROP    drop a context
**    CTBBIND       bind a column variable
**    CTBCLOSE      close a server connection
```

```
**    CTBCONFIG     set or retrieve context properties
**    CTBCMDALLOC   allocate a command
**    CTBCMDDROP    drop a command
**    CTBCOMMAND    initiate remote procedure CALL
**    CTBCONALLOC   allocate a connection
**    CTBCONDROP    drop a connection
**    CTBCONPROPS   alter properties of a connection
**    CTBCONNECT    open a server connection
**    CTBDESCRIBE   return a description of RESULT data
**    CTBDIAG       retrieve SQLCODE messages
**    CTBEXIT       exit client library
**    CTBFETCH      FETCH RESULT data
**    CTBINIT       init client library
**    CTBPARAM      define a command PARAMETER
**    CTBRESULTS    set up RESULT data
**    CTBRESINFO    return RESULT set info
**    CTBSEND       send a request TO the server
**
*****************************************************************

 IDENTIFICATION DIVISION.
 PROGRAM-ID.  SYCTSAT5.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER.  xyz.
 OBJECT-COMPUTER.  xyz.
 DATA DIVISION.
 WORKING-STORAGE SECTION.

*****************************************************************
** CLIENT LIBRARY COBOL COPY BOOK
*****************************************************************
   COPY CTPUBLIC.

*****************************************************************
** CICS BMS DEFINITIONS
*****************************************************************

   COPY SYCTBA5.

*****************************************************************
*  Standard CICS Attribute and Print Control Chararcter List
*****************************************************************

   COPY DFHBMSCA.
```

```
*****************************************************************
** CICS Standard Attention Identifiers Cobol Copy Book
*****************************************************************

   COPY DFHAID.

*****************************************************************
*    CONSTANTS
*****************************************************************
 01  C-N                          PIC X(01) VALUE 'N'.
 01  C-Y                          PIC X(01) VALUE 'Y'.
 01  I1                           PIC S9(9) COMP SYNC VALUE IS 0.
 01  I2                           PIC S9(9) COMP SYNC VALUE IS 0.

 01  MSG-TEXT-1                   PIC X(70) VALUE ' '.
 01  MSG-TEXT-2                   PIC X(70)
                                  VALUE 'Press Clear To Exit'.
 01  PAGE-CNT                     PIC S9(4) COMP VALUE +0.
 01  UTIME                        PIC S9(15) COMP-3.
 01  TMP-DATE                     PIC X(08).
 01  TMP-TIME                     PIC X(08).

 01  MAX-SCREEN-ROWS              PIC S9(4) VALUE +10.

 01  ENTER-DATA-SW                PIC X(01) VALUE 'N'.

*****************************************************************
*    OPEN CLIENT VARIABLES
*****************************************************************
 01  STRLEN                       PIC S9(9) COMP VALUE +0.
 01  OUTLEN                       PIC S9(9) COMP VALUE +0.
 01  RESTYPE                      PIC S9(9) COMP VALUE +0.
 01  NETDRIVER                    PIC S9(9) COMP VALUE +9999.

**---------------------------------------------------------------
** WORK AREAS
**---------------------------------------------------------------

 01  NO-MORE-MSGS-SW              PIC X(01).
     88  NO-MORE-MSGS VALUE 'Y'.

 01  NO-ERRORS-SW                 PIC X(01).
     88  NO-ERRORS    VALUE 'N'.

 01  SWITCHES.
     05  SW-RESULTS                 PIC X(01) VALUE 'Y'.
```

```
           88  NO-MORE-RESULTS VALUE 'N'.
       05  SW-FETCH                  PIC X(01) VALUE 'Y'.
           88  NO-MORE-ROWS VALUE 'N'.
       05  SW-DIAG                   PIC X(01) VALUE 'N'.
           88  DIAG-MSGS-INITIALIZED VALUE 'Y'.

   01  INTERNAL-FIELDS.
       05  I                     PIC S9(9) COMP.
       05  CF-FOUR               PIC S9(9) COMP VALUE +4.
       05  CF-LANG2-SIZE         PIC S9(9) COMP VALUE +120.
       05  DATA-PACKED370        PIC S9(15)V9(3) COMP-3 VALUE +0.

   01  CS-LIB-MISC-FIELDS.
       05  CSL-CMD-HANDLE        PIC S9(9) COMP VALUE +0.
       05  CSL-CON-HANDLE        PIC S9(9) COMP VALUE +0.
       05  CSL-CTX-HANDLE        PIC S9(9) COMP VALUE +0.
       05  CSL-RC                PIC S9(9) COMP VALUE +0.

   01  PROPS-FIELDS.
       05  PF-SERVER             PIC X(30) VALUE IS SPACES.
       05  PF-SERVER-SIZE        PIC S9(9) COMP VALUE +0.
       05  PF-USER               PIC X(08) VALUE IS SPACES.
       05  PF-USER-SIZE          PIC S9(9) COMP VALUE +0.
       05  PF-PWD                PIC X(08) VALUE IS SPACES.
       05  PF-PWD-SIZE           PIC S9(9) COMP VALUE +0.
       05  PF-TRAN               PIC X(08) VALUE IS SPACES.
       05  PF-TRAN-SIZE          PIC S9(9) COMP VALUE +0.
       05  PF-NETDRV             PIC X(08) VALUE IS SPACES.
       05  PF-DRV-SIZE           PIC S9(9) COMP VALUE +0.
       05  PF-STRLEN             PIC S9(9) COMP.
       05  PF-MSGLIMIT           PIC S9(9) COMP.

   01  DIAG-FIELDS.
       05  DG-MSGNO              PIC S9(9) COMP VALUE +1.
       05  DG-NUM-OF-MSGS        PIC S9(9) COMP VALUE +0.

   01  CONFIG-FIELDS.
       05  CF-MAXCONNECT         PIC S9(9) COMP.
       05  CF-OUTLEN             PIC S9(9) COMP.

   01  FETCH-FIELDS.
       05  FF-ROWS-READ          PIC S9(9) COMP.
       05  FF-ROW-NUM            PIC S9(9) COMP VALUE +0.

   01  RESINFO-FIELDS.
       05  RF-NUMDATA            PIC S9(9) COMP.
```

```
    05  RF-NUMDATA-SIZE       PIC S9(9) COMP VALUE +4.

01  OUTPUT-ROW.
    05  OR-COL-PLANEID-CHAR   PIC X(12).
    05  SPACE1                PIC X(01) VALUE ' '.
    05  OR-COL-MILAGE         PIC X(33) VALUE ' '.
    05  SPACE1                PIC X(01) VALUE ' '.
    05  OR-COL-SERVICEDATE    PIC X(21) VALUE ' '.

01  OUTPUT-ROW-STR REDEFINES OUTPUT-ROW PIC X(66).

01  OUTPUT-ROW-SIZE           PIC S9(4) COMP VALUE +66.

01  OUTPUT-ROW2.
    05  OR2-MESG              PIC X(37)
                  VALUE 'The maximum number of connections is '.
    05  OR2-MAXCONNECT        PIC ZZZZ9.
    05  OR2-PERIOD            PIC X(01) VALUE '.'.

01  OUTPUT-ROW-STR2 REDEFINES OUTPUT-ROW2 PIC X(43).

01  OUTPUT-ROW2-SIZE          PIC S9(4) COMP VALUE +43.

01  OUTPUT-ROW4.
    05  OR4-MESG              PIC X(25)
                              VALUE 'The number of columns is '.
    05  OR4-NUMDATA           PIC ZZZZ9.
    05  OR4-PERIOD            PIC X(01)     VALUE '.'.

01  OUTPUT-ROW-STR4 REDEFINES OUTPUT-ROW4 PIC X(31).

01  OUTPUT-ROW4-SIZE          PIC S9(4) COMP VALUE +31.

01  COLUMN-FIELDS.
    05  CF-COL-PLANEID-CHAR   PIC X(12).
    05  CF-COL-MILAGE         PIC X(33) VALUE ' '.
    05  CF-COL-MILAGE-CHAR    PIC X(70) VALUE ' '.
    05  CF-COL-MILAGE-LEN     PIC S9(9) COMP VALUE 0.
    05  CF-COL-SERVICEDATE-CHAR.
        10 CF-COL-DATE-YEAR   PIC X(4).
        10 CF-COL-DATE-SEP1   PIC X(1).
        10 CF-COL-DATE-MM     PIC X(2).
        10 CF-COL-DATE-SEP2   PIC X(1).
        10 CF-COL-DATE-DD     PIC X(2).
        10 SPACE1             PIC X(1) VALUE ' '.
        10 CF-COL-TIME-HH     PIC X(2).
```

```
                10 CF-COL-TIME-SEP1   PIC X(1).
                10 CF-COL-TIME-MM     PIC X(2).
                10 CF-COL-TIME-SEP2   PIC X(1).
                10 CF-COL-TIME-SS     PIC X(2).
            05  CF-COL-LEN            PIC S9(9) COMP VALUE 0.
            05  CF-COL-NULL           PIC S9(9) COMP VALUE +0.
            05  CF-COL-NUMBER         PIC S9(9) COMP VALUE +1.
            05  CF-COL-INDICATOR      PIC S9(4) COMP VALUE +0.

        01  LANG-FIELDS.
            05  CF-LANG1              PIC X(20)
                VALUE 'Wrong SQL statement'.
            05  CF-LANG2              PIC X(115)
                VALUE 'SELECT PLANEID,
MILAGE,CONVERT(CHAR(10),SERVICEDAT02420010
      -         'E,102)+" "+CONVERT(CHAR(8),SERVICEDATE,108) FROM
SYBASE.02430010
      -         'NEWTABLE'.
            05  filler                PIC X(01) VALUE LOW-VALUE.

        01  MSG-FIELDS.
            05  MF-CANCELED           PIC X(16)
                                      VALUE 'Cancel requested'.
            05  MF-CANCELED-SIZE      PIC S9(9) COMP VALUE +16.

        01  DATAFMT.
            05  DF-NAME               PIC X(132).
            05  DF-NAMELEN            PIC S9(9) COMP.
            05  DF-DATATYPE           PIC S9(9) COMP.
            05  DF-FORMAT             PIC S9(9) COMP.
            05  DF-MAXLENGTH          PIC S9(9) COMP.
            05  DF-SCALE              PIC S9(9) COMP.
            05  DF-PRECISION          PIC S9(9) COMP.
            05  DF-STATUS             PIC S9(9) COMP.
            05  DF-COUNT              PIC S9(9) COMP.
            05  DF-USERTYPE           PIC S9(9) COMP.
            05  DF-LOCALE             PIC X(68).

        01  DATAFMT2.
            05  DF2-NAME              PIC X(132).
            05  DF2-NAMELEN           PIC S9(9) COMP.
            05  DF2-DATATYPE          PIC S9(9) COMP.
            05  DF2-FORMAT            PIC S9(9) COMP.
            05  DF2-MAXLENGTH         PIC S9(9) COMP.
            05  DF2-SCALE             PIC S9(9) COMP.
            05  DF2-PRECISION         PIC S9(9) COMP.
```

```
     05  DF2-STATUS            PIC S9(9) COMP.
     05  DF2-COUNT             PIC S9(9) COMP.
     05  DF2-USERTYPE          PIC S9(9) COMP.
     05  DF2-LOCALE            PIC X(68).

 01 DISP-MSG.
    05 TEST-CASE               PIC X(08) VALUE IS 'SYCTSAT5'.
    05 FILLER                  PIC X(01) VALUE IS SPACES.
    05 MSG.
       10 SAMP-LIT             PIC X(05) VALUE IS 'rc = '.
       10 SAMP-RC              PIC -Z9.
       10 FILLER               PIC X(02) VALUE IS ', '.
       10 REST-LIT             PIC X(12) VALUE IS
                                  'Result Type:'.
       10 REST-TYPE            PIC Z(3)9.
       10 FILLER               PIC X(03) VALUE IS SPACES.
       10 MSGSTR               PIC X(40) VALUE IS SPACES.

 01  DISP-MSG-LEN              PIC S9(4) COMP  VALUE IS 65.
 01  MSG-LEN VALUE +0          PIC S9(4) COMP .

*******************************
** Client Message Structure  **
*******************************

 01  CLIENT-MSG.
     05  CM-SEVERITY           PIC S9(9) COMP SYNC.
     05  CM-MSGNO              PIC S9(9) COMP SYNC.
     05  CM-TEXT               PIC X(256).
     05  CM-TEXT-LEN           PIC S9(9) COMP SYNC.
     05  CM-OS-MSGNO           PIC S9(9) COMP SYNC.
     05  CM-OS-MSGTXT          PIC X(256).
     05  CM-OS-MSGTEXT-LEN     PIC S9(9) COMP SYNC.
     05  CM-STATUS             PIC S9(9) COMP.

 01  DISP-CLIENT-MSG-HDR.
     05  CLIENT-MSG-HDR        PIC X(15) VALUE IS
                                   'Client Message:'.

 01  DISP-CLIENT-MSG-1.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  CM-SEVERITY-HDR       PIC X(09) VALUE IS 'Severity:'.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  CM-SEVERITY-DATA      PIC Z(8)9.
     05  CM-STATUS-HDR         PIC X(12) VALUE IS
                                   ',  Status:  '.
```

```
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  CM-STATUS-DATA         PIC Z(8)9.

01  DISP-CLIENT-MSG-2.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  CM-OC-MSGNO-HDR        PIC X(09) VALUE IS 'OC MsgNo:'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  CM-OC-MSGNO-DATA       PIC Z(8)9.

01  DISP-CLIENT-MSG-3.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  CM-OC-MSG-HDR          PIC X(09) VALUE IS 'OC MsgTx:'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  CM-OC-MSG-DATA         PIC X(66).

01  DISP-CLIENT-MSG-3A.
    05  CM-OC-MSG-DATA-1       PIC X(66).
    05  CM-OC-MSG-DATA-2       PIC X(66).
    05  CM-OC-MSG-DATA-3       PIC X(66).
    05  CM-OC-MSG-DATA-4       PIC X(58).

01  DISP-CLIENT-MSG-3B.
    05  FILLER                 PIC X(13) VALUE IS SPACES.
    05  CM-OC-MSG-DATA-X       PIC X(66).

01  DISP-EMPTY-CLIENT-MSG-3.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  CM-OC-MSG-HDR          PIC X(09) VALUE IS 'OC MsgTx:'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  NO-DATA                PIC X(11) VALUE IS 'No Message!'.

01  DISP-CLIENT-MSG-4.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  CM-OS-MSG-HDR          PIC X(09) VALUE IS 'OS MsgNo:'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  CM-OS-MSGNO-DATA       PIC Z(8)9.

01  DISP-CLIENT-MSG-5.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  CM-OS-MSG-HDR          PIC X(09) VALUE IS 'OS MsgTx:'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  CM-OS-MSG-DATA         PIC X(66).

01  DISP-CLIENT-MSG-5A.
    05  CM-OS-MSG-DATA-1       PIC X(66).
    05  CM-OS-MSG-DATA-2       PIC X(66).
```

```
     05  CM-OS-MSG-DATA-3      PIC X(66).
     05  CM-OS-MSG-DATA-4      PIC X(58).

 01  DISP-EMPTY-CLIENT-MSG-5.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  CM-OS-MSG-HDR         PIC X(09) VALUE IS 'OS MsgTx:'.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  NO-DATA               PIC X(11) VALUE IS 'No Message!'.

******************************
** Server Message Structure  **
******************************

 01  SERVER-MSG.
     05  SM-MSGNO              PIC S9(9) COMP.
     05  SM-STATE              PIC S9(9) COMP.
     05  SM-SEV                PIC S9(9) COMP.
     05  SM-TEXT               PIC X(256).
     05  SM-TEXT-LEN           PIC S9(9) COMP.
     05  SM-SVRNAME            PIC X(256).
     05  SM-SVRNAME-LEN        PIC S9(9) COMP.
     05  SM-PROC               PIC X(256).
     05  SM-PROC-LEN           PIC S9(9) COMP.
     05  SM-LINE               PIC S9(9) COMP.
     05  SM-STATUS             PIC S9(9) COMP.

 01  DISP-SERVER-MSG-HDR.
     05  SERVER-MSG-HDR        PIC X(15) VALUE IS
                                    'Server Message:'.

 01  DISP-SERVER-MSG-1.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  SM-MSG-NO-HDR         PIC X(09) VALUE IS
                                    'Message#:'.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  SM-MSG-NO-DATA        PIC Z(8)9.
     05  SM-SEVERITY-HDR       PIC X(12) VALUE IS
                                    ',  Severity:'.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  SM-SEVERITY-DATA      PIC Z(8)9.
     05  SM-STATE-HDR          PIC X(12) VALUE IS
                                    ',  State No:'.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  SM-STATE-DATA         PIC Z(8)9.

 01  DISP-SERVER-MSG-2.
```

```
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-LINE-NO-HDR         PIC X(09) VALUE IS
                                    'Line  No:'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-LINE-NO-DATA        PIC Z(8)9.
    05  SM-STATUS-HDR          PIC X(12) VALUE IS
                                    ',  Status  :'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-STATUS-DATA         PIC Z(8)9.

01  DISP-SERVER-MSG-3.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-SVRNAME-HDR         PIC X(09) VALUE IS 'Serv Nam:'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-SVRNAME-DATA        PIC X(66).
    05  FILLER                 PIC X(03) VALUE IS '...'.

01  DISP-SERVER-MSG-4.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-PROC-ID-HDR         PIC X(09) VALUE IS 'Proc  ID:'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-PROC-ID-DATA        PIC X(66).

01  DISP-SERVER-MSG-5.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-MSG-HDR             PIC X(09) VALUE IS 'Message :'.
    05  FILLER                 PIC X(02) VALUE IS SPACES.
    05  SM-MSG-DATA            PIC X(66).

01  DISP-SERVER-MSG-5A.
    05  SM-MSG-DATA-1          PIC X(66).
    05  SM-MSG-DATA-2          PIC X(66).
    05  SM-MSG-DATA-3          PIC X(66).
    05  SM-MSG-DATA-4          PIC X(58).

01  DISP-SERVER-MSG-5X.
    05  FILLER                 PIC X(13) VALUE IS SPACES.
    05  SM-MSG-DATA-X          PIC X(66).

01  CICS-FIELDS.
    05  CICS-RESPONSE          PIC S9(9) COMP.

01  QUERY-FIELDS.
    05  QF-LEN                 PIC S9(4) COMP VALUE +1.
    05  QF-MAXLEN              PIC S9(4) COMP VALUE +1.
    05  QF-ANSWER              PIC X(01) VALUE IS SPACES.
```

```
 PROCEDURE DIVISION.
**********************

*************************
* CICS Condition Handler *
*************************

     EXEC CICS HANDLE CONDITION MAPFAIL(NO-INPUT)
                                ERROR(ERRORS)
     END-EXEC.

*******************
* CICS Aid Handler *
*******************

     EXEC CICS HANDLE AID ANYKEY(NO-INPUT)
                          CLEAR(GETOUT)
     END-EXEC.

*****************************
*    PROGRAM INITIALIZATION *
*****************************

     MOVE ZERO       TO RESTYPE CSL-RC.

     MOVE C-N    TO NO-MORE-MSGS-SW.
     MOVE C-N    TO NO-ERRORS-SW.
     MOVE C-Y    TO SW-DIAG.

     MOVE LOW-VALUES TO A5PANELO.
     MOVE -1         TO SERVERL.

     COMPUTE PAGE-CNT = PAGE-CNT + 1.

     PERFORM GET-SYSTEM-TIME.

 GET-INPUT-AGAIN.

     PERFORM DISPLAY-INITIAL-SCREEN.

     PERFORM GET-INPUT-DATA.
```

```
*********************************
 *    ALLOCATE A CONTEXT STRUCTURE *
 *********************************

      MOVE ZERO TO CSL-CTX-HANDLE.

      CALL 'CSBCTXAL' USING CS-VERSION-50
                            CSL-RC
                            CSL-CTX-HANDLE.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
           MOVE SPACES TO MSGSTR
           STRING 'CSBCTXAL failed' DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
      END-IF.

 *********************************
 * INTITIALIZE THE CLIENT-LIBRARY *
 *********************************

      CALL 'CTBINIT' USING CSL-CTX-HANDLE
                           CSL-RC
                           CS-VERSION-50.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBINIT failed' DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
      END-IF.

      PERFORM PROCESS-MESSAGES.

      PERFORM QUIT-CLIENT-LIBRARY.

      GOBACK.

 *============================================================
 *==                                                        ==
 *== Subroutine to get system date/time                     ==
 *==                                                        ==
 *============================================================
  GET-SYSTEM-TIME.
```

```
*----------------

     EXEC CICS ASKTIME
               ABSTIME(UTIME)
     END-EXEC.

     EXEC CICS FORMATTIME
               ABSTIME(UTIME)
               DATESEP('/')
               MMDDYY(TMP-DATE)
               TIME(TMP-TIME)
               TIMESEP
     END-EXEC.

*============================================================
*==                                                        ==
*== Subroutine to display SYT5 initial screen              ==
*==                                                        ==
*============================================================
 DISPLAY-INITIAL-SCREEN.
*----------------------

     MOVE TMP-DATE   TO SDATEO.
     MOVE TMP-TIME   TO STIMEO.
     MOVE 'SYCTSAT5' TO PROGNMO.

     MOVE PAGE-CNT   TO SPAGEO.
     MOVE MSG-TEXT-1 TO MSG1O.
     MOVE MSG-TEXT-2 TO MSG2O.

     EXEC CICS SEND MAP('A5PANEL')
                    MAPSET('SYCTBA5')
                    CURSOR
                    FRSET
                    ERASE
                    FREEKB
     END-EXEC.

*============================================================
*==                                                        ==
*== Subroutine to get input data                           ==
*==                                                        ==
*============================================================
 GET-INPUT-DATA.
*---------------
```

```
EXEC CICS RECEIVE MAP('A5PANEL')
               MAPSET('SYCTBA5')
               ASIS
END-EXEC.

IF SERVERL = ZERO
  THEN
    IF PF-SERVER = SPACES
      THEN
        MOVE 'Please Enter Server Name' TO MSG-TEXT-1
        MOVE -1                         TO SERVERL
        MOVE C-Y                        TO ENTER-DATA-SW
    END-IF
  ELSE
    MOVE SERVERI   TO PF-SERVER
    MOVE SERVERL   TO PF-SERVER-SIZE
END-IF.

IF USERL = ZERO
  THEN
    IF PF-USER = SPACES
      THEN
        MOVE 'Please Enter User-ID' TO MSG-TEXT-1
        MOVE -1                     TO USERL
        MOVE C-Y                    TO ENTER-DATA-SW
    END-IF
  ELSE
    MOVE USERI   TO PF-USER
    MOVE USERL   TO PF-USER-SIZE
    MOVE PF-USER TO USERO
END-IF.

IF PSWDL NOT EQUAL ZERO
  THEN
    MOVE PSWDI TO PF-PWD
    MOVE PSWDL TO PF-PWD-SIZE
END-IF.

IF TRANL NOT EQUAL ZERO
  THEN
    MOVE TRANI TO PF-TRAN
    MOVE TRANL TO PF-TRAN-SIZE
END-IF.

IF NETDRVL NOT EQUAL ZERO
  THEN
```

```
           MOVE NETDRVI TO PF-NETDRV
           MOVE NETDRVL TO PF-DRV-SIZE
      END-IF.

      IF ENTER-DATA-SW = C-Y
        THEN
           MOVE C-N TO ENTER-DATA-SW
           PERFORM DISPLAY-INITIAL-SCREEN
           PERFORM GET-INPUT-DATA
      END-IF.

*============================================================
*==                                                        ==
*== Subroutine to process input data                       ==
*==                                                        ==
*============================================================
 PROCESS-MESSAGES.

********************************
* ALLOCATE A CONNECTION HANDLE. *
********************************

      MOVE ZERO TO CSL-CON-HANDLE.

      CALL 'CTBCONAL' USING CSL-CTX-HANDLE
                            CSL-RC
                            CSL-CON-HANDLE.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBCONAL failed' DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
      END-IF.

******************
* SET THE USER ID *
******************

      CALL 'CTBCONPR' USING CSL-CON-HANDLE
                            CSL-RC
                            CS-SET
                            CS-USERNAME
                            PF-USER
                            PF-USER-SIZE
```

```
                          CS-FALSE
                          OUTLEN.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBCONPR for user-id failed' DELIMITED BY SIZE
                                               INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
     END-IF.

*********************
* SET THE PASSWORD *
*********************

     CALL 'CTBCONPR' USING CSL-CON-HANDLE
                          CSL-RC
                          CS-SET
                          CS-PASSWORD
                          PF-PWD
                          PF-PWD-SIZE
                          CS-FALSE
                          OUTLEN.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBCONPR for password failed' DELIMITED BY SIZE
                                                INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
     END-IF.

*********************
* SET THE TRAN NAME *
*********************

     IF PF-TRAN-SIZE IS NOT EQUAL TO ZEROES THEN

     CALL 'CTBCONPR' USING CSL-CON-HANDLE
                          CSL-RC
                          CS-SET
                          CS-TRANSACTION-NAME
                          PF-TRAN
                          PF-TRAN-SIZE
```

```
                              CS-FALSE
                              OUTLEN

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBCONPR for TRANname failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
        END-IF

      END-IF.

******************************
* SET THE NET DRIVER PROPERTY *
******************************

      IF PF-NETDRV = SPACES OR PF-NETDRV = 'LU62'         X
                              OR PF-NETDRV = 'lu62'
          MOVE CS-LU62 TO NETDRIVER
      ELSE
        IF PF-NETDRV = 'IBMTCPIP' OR PF-NETDRV = 'ibmtcpip'
          MOVE CS-TCPIP TO NETDRIVER
      ELSE
        IF PF-NETDRV = 'INTERLIN' OR PF-NETDRV = 'interlin'
          MOVE CS-INTERLINK TO NETDRIVER
      ELSE
        IF PF-NETDRV = 'CPIC' OR PF-NETDRV = 'cpic'
          MOVE CS-NCPIC TO NETDRIVER
      END-IF.

      IF PF-DRV-SIZE IS NOT EQUAL TO ZEROES THEN

          CALL 'CTBCONPR' USING CSL-CON-HANDLE
                                CSL-RC
                                CS-SET
                                CS-NET-DRIVER
                                NETDRIVER
                                CS-UNUSED
                                CS-FALSE
                                OUTLEN

          IF CSL-RC NOT EQUAL CS-SUCCEED
            THEN
               MOVE SPACES TO MSGSTR
```

```
               STRING 'CTBCONPR for network driver failed'
                      DELIMITED BY SIZE INTO MSGSTR
               PERFORM PRINT-MSG
               PERFORM ALL-DONE
             END-IF

     END-IF.

**********************************
* SETUP retrieval of All Messages *
**********************************

     CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                          CSL-RC,
                          CS-UNUSED,
                          CS-INIT,
                          CS-ALLMSG-TYPE,
                          CS-UNUSED,
                          CS-UNUSED.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBDIAG CS-INIT failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

*****************************************
* set the upper limit of number of messages *
*****************************************

     MOVE 5 TO PF-MSGLIMIT.

     CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                          CSL-RC,
                          CS-UNUSED,
                          CS-MSGLIMIT,
                          CS-ALLMSG-TYPE,
                          CS-UNUSED,
                          PF-MSGLIMIT.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
```

```
           STRING 'CTBDIAG CS-MSGLIMIT failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
       END-IF.

*********************************************
* CONNECT TO THE SERVER OR THE IMS/CICS REGION *
*********************************************

       CALL 'CTBCONNE' USING CSL-CON-HANDLE
                             CSL-RC
                             PF-SERVER
                             PF-SERVER-SIZE
                             CS-FALSE.

       IF CSL-RC NOT EQUAL CS-SUCCEED
         THEN
            MOVE SPACES TO MSGSTR
            STRING 'CTBCONNE failed' DELIMITED BY SIZE INTO MSGSTR
            PERFORM PRINT-MSG
            PERFORM ALL-DONE
       END-IF

       IF NO-ERRORS
         THEN
           PERFORM SEND-COMMAND
       END-IF

************************************
* PROCESS THE RESULTS OF THE COMMAND *
************************************

       IF NO-ERRORS
         THEN
           PERFORM RESULTS-PROCESSING UNTIL NO-MORE-RESULTS
           PERFORM CLOSE-CONNECTION
       END-IF.

 PROCESS-MESSAGES-EXIT.
       EXIT.

*=========================================================
*==                                                     ==
*== Subroutine to allocate, send, and process commands ==
*==                                                     ==
```

```
*=========================================================
 SEND-COMMAND.

*---------------------------------------------------------------
*  find out what the maximum number of connections is
*---------------------------------------------------------------
     CALL 'CTBCONFI'  USING CSL-CTX-HANDLE,
                            CSL-RC,
                            CS-GET,
                            CS-MAX-CONNECT,
                            CF-MAXCONNECT,
                            CF-FOUR,
                            CS-FALSE,
                            CF-OUTLEN.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBCONFI CS-GET failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

*---------------------------------------------------------------
*  allocate a command handle
*---------------------------------------------------------------

     CALL 'CTBCMDAL' USING CSL-CON-HANDLE,
                           CSL-RC,
                           CSL-CMD-HANDLE.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBCMDAL failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

*---------------------------------------------------------------
*    prepare the language request
*---------------------------------------------------------------

     MOVE CF-LANG2-SIZE TO PF-STRLEN.
```

```
     CALL 'CTBCOMMA' USING CSL-CMD-HANDLE,
                           CSL-RC,
                           CS-LANG-CMD,
                           CF-LANG2,
                           PF-STRLEN,
                           CS-UNUSED.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBCOMMA CS-LANG-CMD failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

*-------------------------------------------------------------
*    send the language request
*-------------------------------------------------------------
     CALL 'CTBSEND' USING CSL-CMD-HANDLE,
                          CSL-RC.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBSEND failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

 SEND-COMMAND-EXIT.
     EXIT.

*============================================================
*==                                                        ==
*== Subroutine to process result                           ==
*==                                                        ==
*============================================================
 RESULTS-PROCESSING.

**************************
* SET UP THE RESULTS DATA *
**************************
```

```
     CALL 'CTBRESUL' USING CSL-CMD-HANDLE
                           CSL-RC
                           RESTYPE.

************************************************
* DETERMINE THE OUTCOME OF THE COMMAND EXECUTION *
************************************************

     EVALUATE CSL-RC

         WHEN CS-SUCCEED

*****************************************************************
* DETERMINE THE TYPE OF RESULT RETURNED BY THE CURRENT REQUEST *
*****************************************************************

             EVALUATE RESTYPE

***********************
* PROCESS ROW RESULTS *
***********************

               WHEN CS-ROW-RESULT
                 PERFORM RESULT-ROW-PROCESSING
                 MOVE 'Y' TO SW-FETCH
                 PERFORM FETCH-ROW-PROCESSING UNTIL NO-MORE-ROWS

************************************************************
* PROCESS PARAMETER RESULTS - THERE SHOULD BE NO PARAMETERS *
* TO PROCESS                                                *
************************************************************

               WHEN CS-PARAM-RESULT
                 MOVE 'Y' TO SW-FETCH

************************************************************
* PROCESS STATUS RESULTS - THE STORED PROCEDURE STATUS RESULT *
* WILL NOT BE PROCESSED IN THIS EXAMPLE                       *
************************************************************

               WHEN CS-STATUS-RESULT
                 MOVE 'Y' TO SW-FETCH

************************************************************
* PRINT AN ERROR MESSAGE IF THE SERVER ENCOUNTERED AN ERROR *
* WHILE EXECUTING THE REQUEST                               *
```

```
*************************************************************

               WHEN CS-CMD-FAIL
                 STRING
                    'CTBRESUL returned CS-CMD-FAIL restype'
                    DELIMITED BY SIZE INTO MSGSTR
                 PERFORM PRINT-MSG

*****************************************************************
* PRINT A MESSAGE FOR SUCCESSFUL COMMANDS THAT RETURNED NO DATA *
* (OPTIONAL)                                                    *
*****************************************************************
               WHEN CS-CMD-SUCCEED
                 STRING
                    'CTBRESUL returned CS-CMD-SUCCEED restype'
                    DELIMITED BY SIZE INTO MSGSTR

*********************************************************
* PRINT A MESSAGE FOR REQUESTS THAT HAVE BEEN PROCESSED *
* SUCCESSFULLY (OPTIONAL)                               *
*********************************************************

               WHEN CS-CMD-DONE
                 STRING 'CTBRESUL returned CS-CMD-DONE restype'
                             DELIMITED BY SIZE INTO MSGSTR

               WHEN OTHER
                  STRING 'CTBRESUL returned UNKNOWN restype'
                             DELIMITED BY SIZE INTO MSGSTR
                  PERFORM PRINT-MSG
                  MOVE 'N' TO SW-RESULTS

            END-EVALUATE

*******************************************************
* PRINT AN ERROR MESSAGE IF THE CTBRESULTS CALL FAILED *
*******************************************************

         WHEN CS-FAIL
           MOVE 'N' TO SW-RESULTS
           STRING 'CTBRESUL returned CS-FAIL ret-code'
                     DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG

*********************************************************
* DROP OUT OF THE RESULTS LOOP IF NO MORE RESULT SETS ARE   *
```

```
* AVAILABLE FOR PROCESSING OR IF THE RESULTS WERE CANCELLED *
************************************************************

         WHEN CS-END-RESULTS
           MOVE 'N' TO SW-RESULTS

         WHEN CS-CANCELLED
           MOVE 'N' TO SW-RESULTS

         WHEN OTHER
           MOVE 'N' TO SW-RESULTS
           STRING 'CTBRESUL returned UNKNOWN ret-code'
                     DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG

     END-EVALUATE.

     MOVE 0 TO RESTYPE.

 RESULTS-PROCESSING-EXIT.
     EXIT.

*=========================================================
*==                                                     ==
*== Subroutine to process result rows                   ==
*==                                                     ==
*=========================================================
 RESULT-ROW-PROCESSING.

     CALL 'CTBRESIN' USING CSL-CMD-HANDLE,
                           CSL-RC,
                           CS-NUMDATA,
                           RF-NUMDATA,
                           RF-NUMDATA-SIZE,
                           CF-COL-LEN.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBRESINFO failed' DELIMITED BY SIZE
                                                 INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

     COMPUTE FF-ROW-NUM = FF-ROW-NUM + 1.
```

```
*********************************
* display number of connections *
*********************************

     MOVE CF-MAXCONNECT   TO OR2-MAXCONNECT.
     MOVE OUTPUT-ROW-STR2 TO RSLTNO(FF-ROW-NUM).
     COMPUTE FF-ROW-NUM = FF-ROW-NUM + 2.

*********************************
* display the number of columns *
*********************************

     MOVE RF-NUMDATA      TO OR4-NUMDATA.
     MOVE OUTPUT-ROW-STR4 TO RSLTNO(FF-ROW-NUM).

     IF RF-NUMDATA NOT EQUAL 3
       THEN
         STRING 'CTBRESINFO returned wrong # of parms' DELIMITED
                                          BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

     COMPUTE FF-ROW-NUM = FF-ROW-NUM + 2.

**-------------------------------------------------------------
**   Setup column headings
**-------------------------------------------------------------

     MOVE 'PLANE ID               MILAGE               Serv -
          'ice Date'  TO RSLTNO(FF-ROW-NUM).
     COMPUTE FF-ROW-NUM = FF-ROW-NUM + 1.
     MOVE '============ ======================================== -
       '================'      TO RSLTNO(FF-ROW-NUM).
     PERFORM BIND-COLUMNS
        VARYING I FROM 1 BY 1
           UNTIL I IS GREATER THAN RF-NUMDATA.

 RESULT-ROW-PROCESSING-EXIT.
     EXIT.

*=========================================================
*==                                                     ==
*== Subroutine to bind each data                        ==
*==                                                     ==
```

```
*=========================================================
 BIND-COLUMNS.

      CALL 'CTBDESCR' USING CSL-CMD-HANDLE,
                            CSL-RC,
                            I,
                            DATAFMT.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBDESCR failed'
                 DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.

 **-------------------------------------------------------
 **   We need to bind the data to program variables.
 **   We don't care about the indicator variable
 **   so we'll pass NULL for that parameter in OC-BIND().
 **-------------------------------------------------------

 ******************
 * ROWs per FETCH *
 ******************
       MOVE 1 TO DF-COUNT

       EVALUATE DF-DATATYPE ALSO I

         WHEN CS-DECIMAL-TYPE ALSO 2


**-------------------------------------------------------
 **   The maximum length should be the precision  of the
 **   decimal item + 2. One byte for sign and one for the
 **   decimal point.
 **-------------------------------------------------------
           MOVE DF-PRECISION
                TO DF-MAXLENGTH
           ADD 2 TO DF-MAXLENGTH
           MOVE CS-CHAR-TYPE TO DF-DATATYPE
           CALL 'CTBBIND' USING CSL-CMD-HANDLE,
                                CSL-RC,
                                I,
                                DATAFMT,
```

```
                        CF-COL-MILAGE-CHAR,
                        DF-MAXLENGTH,
                        CS-PARAM-NOTNULL,
                        CF-COL-INDICATOR,
                        CS-PARAM-NULL

    IF CSL-RC NOT EQUAL CS-SUCCEED
      THEN
        MOVE SPACES TO MSGSTR
        STRING 'CTBBIND CS-DECIMAL-TYPE Filed' DELIMITED
                                    BY SIZE INTO MSGSTR
        PERFORM PRINT-MSG
        PERFORM ALL-DONE
    END-IF

  WHEN CS-VARCHAR-TYPE ALSO 1
    MOVE CS-CHAR-TYPE TO DF-DATATYPE
    MOVE LENGTH OF CF-COL-PLANEID-CHAR TO DF-MAXLENGTH

    CALL 'CTBBIND' USING CSL-CMD-HANDLE,
                        CSL-RC,
                        I,
                        DATAFMT,
                        CF-COL-PLANEID-CHAR,
                        CF-COL-LEN,
                        CS-PARAM-NOTNULL,
                        CF-COL-INDICATOR,
                        CS-PARAM-NULL

    IF CSL-RC NOT EQUAL CS-SUCCEED
      THEN
        MOVE SPACES TO MSGSTR
        STRING 'CTBBIND CS-VARCHAR-TYPE failed' DELIMITED
                                    BY SIZE INTO MSGSTR
        PERFORM PRINT-MSG
        PERFORM ALL-DONE
    END-IF
  WHEN CS-VARCHAR-TYPE ALSO 3
    MOVE CS-CHAR-TYPE TO DF-DATATYPE
    MOVE LENGTH OF CF-COL-SERVICEDATE-CHAR
         TO DF-MAXLENGTH

    CALL 'CTBBIND' USING CSL-CMD-HANDLE,
                        CSL-RC,
                        I,
                        DATAFMT,
```

```
                                    CF-COL-SERVICEDATE-CHAR,
                                    CF-COL-LEN,
                                    CS-PARAM-NOTNULL,
                                    CF-COL-INDICATOR,
                                    CS-PARAM-NULL

            IF CSL-RC NOT EQUAL CS-SUCCEED
              THEN
                MOVE SPACES TO MSGSTR
                STRING 'CTBBIND CS-DATETIME-TYPE failed' DELIMITED
                                              BY SIZE INTO MSGSTR
                PERFORM PRINT-MSG
                PERFORM ALL-DONE
            END-IF.

 BIND-COLUMNS-EXIT.
     EXIT.

*=========================================================
*==                                                     ==
*== Subroutine to fetch row processing                  ==
*==                                                     ==
*=========================================================
 FETCH-ROW-PROCESSING.

     CALL 'CTBFETCH' USING CSL-CMD-HANDLE,
                           CSL-RC,
                           CS-UNUSED,
                           CS-UNUSED,
                           CS-UNUSED,
                           FF-ROWS-READ.

     EVALUATE CSL-RC

         WHEN CS-SUCCEED
         MOVE 'Y'              TO SW-FETCH

         COMPUTE FF-ROW-NUM = FF-ROW-NUM + 1
************************************
* save ROW RESULTS for later display *
************************************
         MOVE CF-COL-PLANEID-CHAR TO
              OR-COL-PLANEID-CHAR
         MOVE CF-COL-MILAGE-CHAR   TO
              OR-COL-MILAGE
         MOVE '-' TO CF-COL-DATE-SEP1, CF-COL-DATE-SEP2
```

```
           MOVE '.' TO CF-COL-TIME-SEP1, CF-COL-TIME-SEP2
           MOVE CF-COL-SERVICEDATE-CHAR TO
                OR-COL-SERVICEDATE
           IF FF-ROW-NUM > MAX-SCREEN-ROWS
             THEN
               STRING 'Please press enter for more data.'
                      DELIMITED BY SIZE INTO MSG1O
               PERFORM DISP-DATA
               PERFORM CLEAR-SCREEN-DATA
                       VARYING I2 FROM 1 BY 1
                       UNTIL I2 > MAX-SCREEN-ROWS
               MOVE 1      TO FF-ROW-NUM
      **-------------------------------------------------------------
      **   Setup column headings
      **-------------------------------------------------------------
               MOVE '  Plane ID               Milage             -
                    '    Service  Date     '
               TO RSLTNO(FF-ROW-NUM)
               COMPUTE FF-ROW-NUM = FF-ROW-NUM + 1
               MOVE '============ ================================ -
                    '====================='
                    TO  RSLTNO(FF-ROW-NUM)
                    COMPUTE FF-ROW-NUM = FF-ROW-NUM + 1
                END-IF

                MOVE OUTPUT-ROW-STR TO RSLTNO(FF-ROW-NUM)

                MOVE SPACES         TO CF-COL-PLANEID-CHAR

           WHEN CS-END-DATA
                MOVE SPACES TO MSG1O
                MOVE 'N'    TO SW-FETCH
                STRING 'All rows processing completed!'
                       DELIMITED BY SIZE INTO MSG1O
                PERFORM DISP-DATA

           WHEN CS-FAIL
                MOVE 'N'    TO SW-FETCH
                MOVE SPACES TO MSGSTR
                STRING 'CTBFETCH returned CS-FAIL ret-code'
                       DELIMITED BY SIZE INTO MSGSTR
                PERFORM PRINT-MSG

           WHEN CS-ROW-FAIL
                MOVE 'N'    TO SW-FETCH
                MOVE SPACES TO MSGSTR
```

```
               STRING 'CTBFETCH returned CS-ROW-FAIL ret-code'
                              DELIMITED BY SIZE INTO MSGSTR
               PERFORM PRINT-MSG

          WHEN CS-CANCELLED
               MOVE 'N'          TO SW-FETCH
               MOVE MF-CANCELED TO MSG1O
               PERFORM PRINT-MSG

          WHEN OTHER
               MOVE 'N'    TO SW-FETCH
               MOVE SPACES TO MSGSTR
               STRING 'CTBFETCH returned UNKNOWN ret-code'
                              DELIMITED BY SIZE INTO MSGSTR
               PERFORM PRINT-MSG

     END-EVALUATE.

 FETCH-ROW-PROCESSING-EXIT.
     EXIT.

*=========================================================
*==                                                     ==
*== Subroutine to tell CICS to send output messages    ==
*==                                                     ==
*=========================================================
 DISP-DATA.

***********************************************************
* PRINT ALL THE RETURNED ROWS FROM THE STORED PROCEDURE *
***********************************************************

     MOVE TMP-DATE   TO SDATEO.
     MOVE TMP-TIME   TO STIMEO.
     MOVE 'SYCTSAT5' TO PROGNMO.
     MOVE PAGE-CNT   TO SPAGEO.

     MOVE DFHBMPRO   TO SERVERA.
     MOVE PF-SERVER  TO SERVERO.

     MOVE DFHBMPRO   TO USERA.
     MOVE PF-USER    TO USERO.

     MOVE DFHBMPRO   TO NETDRVA.
     MOVE PF-NETDRV  TO NETDRVO.
```

```
     MOVE DFHBMDAR   TO PSWDA.
     MOVE PF-PWD    TO PSWDO.
     MOVE MSG-TEXT-2 TO MSG2O.

*******************
* DISPLAY THE DATA *
*******************

     EXEC CICS SEND MAP('A5PANEL')
                    MAPSET('SYCTBA5')
                    CURSOR
                    FRSET
                    ERASE
                    FREEKB
     END-EXEC.

     EXEC CICS RECEIVE INTO(QF-ANSWER)
                       LENGTH(QF-LEN)
                       MAXLENGTH(QF-MAXLEN)
                       RESP(CICS-RESPONSE)
     END-EXEC.

 DISP-DATA-EXIT.
     EXIT.

*============================================================
*==                                                        ==
*== Subroutine to print output messages.                   ==
*==                                                        ==
*============================================================
 PRINT-MSG.

     MOVE CSL-RC  TO SAMP-RC.
     MOVE RESTYPE TO REST-TYPE.

     IF DIAG-MSGS-INITIALIZED
       THEN
         PERFORM GET-DIAG-MESSAGES
     END-IF.

***********************
* DISPLAY THE MESSAGE *
***********************

     MOVE DISP-MSG TO MSG1O.
```

```
      IF NO-ERRORS
        THEN
          PERFORM DISP-DATA.

      MOVE C-Y    TO NO-ERRORS-SW.
      MOVE SPACES TO MSGSTR.
      MOVE SPACES TO MSG1O.
      MOVE ZERO   TO SAMP-RC.
      MOVE ZERO   TO REST-TYPE.

  PRINT-MSG-EXIT.
      EXIT.



*=========================================================
 *==                                                    ==
 *== Subroutine to drop and to deallocate all handlers, ==
 *== to close server connection and exit client library ==
 *==                                                    ==
 *========================================================
  ALL-DONE.

      PERFORM CLOSE-CONNECTION.
      PERFORM QUIT-CLIENT-LIBRARY.
      STOP RUN.

  ALL-DONE-EXIT.
      EXIT.

 *============================================================
 *==                                                        ==
 *== Subroutine to perform drop command handler, close     ==
 *== server connection, and deallocate Connection Handler. ==
 *==                                                        ==
 *============================================================
  CLOSE-CONNECTION.

 ***************************
 * DROP THE COMMAND HANDLE *
 ***************************

      CALL 'CTBCMDDR' USING CSL-CMD-HANDLE
                            CSL-RC.

      IF CSL-RC = CS-FAIL
```

```
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBCMDDR failed' DELIMITED BY
                  SIZE INTO MSGSTR
          PERFORM PRINT-MSG
      END-IF.

*******************************
* CLOSE THE SERVER CONNECTION *
*******************************

      CALL 'CTBCLOSE' USING CSL-CON-HANDLE
                            CSL-RC
                            CS-UNUSED.

      IF CSL-RC = CS-FAIL
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBCLOSE failed' DELIMITED BY
                  SIZE INTO MSGSTR
          PERFORM PRINT-MSG
      END-IF.

***********************************
* DE-ALLOCATE THE CONNECTION HANDLE *
***********************************

      CALL 'CTBCONDR' USING CSL-CON-HANDLE
                            CSL-RC.

      IF CSL-RC = CS-FAIL
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBCONDR failed' DELIMITED BY
                  SIZE INTO MSGSTR
          PERFORM PRINT-MSG
      END-IF.

 CLOSE-CONNECTION-EXIT.
      EXIT.

*================================================
*==                                            ==
*== Subroutine to perform exit client library and ==
*== deallocate context structure.              ==
*==                                            ==
```

```
*=================================================
 QUIT-CLIENT-LIBRARY.

***************************
* EXIT THE CLIENT LIBRARY *
***************************

     CALL 'CTBEXIT' USING CSL-CTX-HANDLE
                          CSL-RC
                          CS-UNUSED.
     IF CSL-RC = CS-FAIL
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBEXIT failed' DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
     END-IF.

***********************************
* DE-ALLOCATE THE CONTEXT STRUCTURE *
***********************************

     CALL 'CSBCTXDR' USING CSL-CTX-HANDLE
                           CSL-RC.

     IF CSL-RC = CS-FAIL
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CSBCTXDR failed' DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
     END-IF.

 QUIT-CLIENT-LIBRARY-EXIT.
     EXIT.

*=====================================================
*==                                                 ==
*== Subroutine to retrieve any diagnostic messages ==
*==                                                 ==
*=====================================================
 GET-DIAG-MESSAGES.

***********************************
* Disable calls to this subroutine *
***********************************

     MOVE 'N' TO SW-DIAG.
```

```
*****************************
* First, get client messages *
*****************************

     CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                          CSL-RC,
                          CS-UNUSED,
                          CS-STATUS,
                          CS-CLIENTMSG-TYPE,
                          CS-UNUSED,
                          DG-NUM-OF-MSGS.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBDIAG CS-STATUS CS-CLIENTMSG-TYP fail'
                        DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
       ELSE
         IF DG-NUM-OF-MSGS > 0
           THEN
             PERFORM RETRIEVE-CLIENT-MSGS
                 VARYING I FROM 1 BY 1
                     UNTIL I IS GREATER THAN DG-NUM-OF-MSGS
         END-IF
     END-IF.

*****************************
* Then, get server messages *
*****************************

     CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                          CSL-RC,
                          CS-UNUSED,
                          CS-STATUS,
                          CS-SERVERMSG-TYPE,
                          CS-UNUSED,
                          DG-NUM-OF-MSGS.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         STRING 'CTBDIAG CS-STATUS CS-SERVERMSG-TYP fail'
                        DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
```

```
          PERFORM ALL-DONE
        ELSE
          IF DG-NUM-OF-MSGS > 0
            THEN
              PERFORM RETRIEVE-SERVER-MSGS
                  VARYING I FROM 1 BY 1
                      UNTIL I IS GREATER THAN DG-NUM-OF-MSGS
          END-IF
      END-IF.

 GET-DIAG-MESSAGES-EXIT.
      EXIT.

*============================================================*
*==                                                        ==
*== Subroutine to retrieve diagnostic messages from client ==
*==                                                        ==
*============================================================*
 RETRIEVE-CLIENT-MSGS.

      MOVE 1 TO I1.

      CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                           CSL-RC,
                           CS-UNUSED,
                           CS-GET,
                           CS-CLIENTMSG-TYPE,
                           DG-MSGNO,
                           CLIENT-MSG.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBDIAG CS-GET CS-CLIENTMSG-TYPE failed'
                          DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.

************************
* display message text *
************************

      MOVE DISP-CLIENT-MSG-HDR TO RSLTNO( I1 ).
      MOVE 3 TO I1.
```

```
MOVE CM-SEVERITY       TO CM-SEVERITY-DATA.
MOVE CM-STATUS         TO CM-STATUS-DATA.
MOVE DISP-CLIENT-MSG-1 TO RSLTNO( I1 ).
COMPUTE I1 EQUAL I1 + 1

MOVE CM-MSGNO          TO CM-OC-MSGNO-DATA.
MOVE DISP-CLIENT-MSG-2 TO RSLTNO( I1 ).
COMPUTE I1 EQUAL I1 + 1

IF CM-MSGNO NOT EQUAL 0
  THEN
    MOVE SPACES            TO CM-OC-MSG-DATA
    MOVE CM-TEXT           TO CM-OC-MSG-DATA
    MOVE CM-TEXT           TO DISP-CLIENT-MSG-3A
    MOVE DISP-CLIENT-MSG-3 TO RSLTNO( I1 )
    COMPUTE I1 EQUAL I1 + 1
    IF CM-TEXT-LEN > 66
      THEN
        MOVE CM-OC-MSG-DATA-2   TO CM-OC-MSG-DATA-X
        MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
        COMPUTE I1 EQUAL I1 + 1
        IF CM-TEXT-LEN > 132
          THEN
            MOVE SPACES             TO CM-OC-MSG-DATA-X
            MOVE CM-OC-MSG-DATA-3   TO CM-OC-MSG-DATA-X
            MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
            COMPUTE I1 EQUAL I1 + 1
            IF CM-TEXT-LEN > 198
              THEN
                MOVE SPACES             TO CM-OC-MSG-DATA-X
                MOVE CM-OC-MSG-DATA-4   TO CM-OC-MSG-DATA-X
                MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
                COMPUTE I1 EQUAL I1 + 1
            END-IF
        END-IF
      END-IF
  ELSE
    MOVE DISP-EMPTY-CLIENT-MSG-3 TO RSLTNO( I1 )
    COMPUTE I1 EQUAL I1 + 1
END-IF.

MOVE CM-OS-MSGNO       TO CM-OS-MSGNO-DATA.
MOVE DISP-CLIENT-MSG-4 TO RSLTNO( I1 ).
COMPUTE I1 EQUAL I1 + 1

IF CM-OS-MSGNO NOT EQUAL 0
```

```
        THEN
          MOVE SPACES            TO CM-OS-MSG-DATA
          MOVE CM-OS-MSGTXT      TO CM-OS-MSG-DATA
          MOVE SPACES            TO DISP-CLIENT-MSG-5A
          MOVE CM-OS-MSGTXT      TO DISP-CLIENT-MSG-5A
          MOVE DISP-CLIENT-MSG-5 TO RSLTNO( I1 )
          COMPUTE I1 EQUAL I1 + 1
          IF CM-OS-MSGTEXT-LEN > 66
            THEN
              MOVE SPACES             TO CM-OC-MSG-DATA-X
              MOVE CM-OS-MSG-DATA-2   TO CM-OC-MSG-DATA-X
              MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
              COMPUTE I1 EQUAL I1 + 1
              IF CM-OS-MSGTEXT-LEN > 132
                THEN
                  MOVE SPACES             TO CM-OC-MSG-DATA-X
                  MOVE CM-OS-MSG-DATA-3   TO CM-OC-MSG-DATA-X
                  MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
                  COMPUTE I1 EQUAL I1 + 1
                  IF CM-OS-MSGTEXT-LEN > 198
                    THEN
                      MOVE SPACES             TO CM-OC-MSG-DATA-X
                      MOVE CM-OS-MSG-DATA-4   TO CM-OC-MSG-DATA-X
                      MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
                      COMPUTE I1 EQUAL I1 + 1
                  END-IF
              END-IF
          END-IF
        ELSE
          MOVE DISP-EMPTY-CLIENT-MSG-5 TO RSLTNO( I1 )
          COMPUTE I1 EQUAL I1 + 1
      END-IF.

 RETRIEVE-CLIENT-MSGS-EXIT.
      EXIT.

*============================================================
*==                                                        ==
*== Subroutine to retrieve diagnostic messages from server ==
*==                                                        ==
*============================================================
 RETRIEVE-SERVER-MSGS.

      CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                           CSL-RC,
                           CS-UNUSED,
```

```
                          CS-GET,
                          CS-SERVERMSG-TYPE,
                          DG-MSGNO,
                          SERVER-MSG.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBDIAG CS-GET CS-SERVERMSG-TYPE failed'
                         DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

************************
* display message text *
************************

     MOVE SM-MSGNO   TO SM-MSG-NO-DATA.
     MOVE SM-SEV     TO SM-SEVERITY-DATA.
     MOVE SM-STATE   TO SM-STATE-DATA.

     MOVE SM-LINE    TO SM-LINE-NO-DATA.
     MOVE SM-STATUS  TO SM-STATUS-DATA.

     MOVE SPACES     TO SM-SVRNAME-DATA.
     MOVE SM-SVRNAME TO SM-SVRNAME-DATA.

     MOVE SPACES     TO SM-PROC-ID-DATA.
     MOVE SM-PROC    TO SM-PROC-ID-DATA.

     MOVE SPACES     TO SM-MSG-DATA.
     MOVE SM-TEXT    TO SM-MSG-DATA.

     MOVE SPACES     TO DISP-SERVER-MSG-5A.
     MOVE SM-TEXT    TO DISP-SERVER-MSG-5A.


     MOVE DISP-SERVER-MSG-HDR TO RSLTNO (1).
     MOVE DISP-SERVER-MSG-1   TO RSLTNO (3).
     MOVE DISP-SERVER-MSG-2   TO RSLTNO (4).
     MOVE DISP-SERVER-MSG-3   TO RSLTNO (5).
     MOVE DISP-SERVER-MSG-4   TO RSLTNO (6).

     MOVE DISP-SERVER-MSG-5   TO RSLTNO (7).
     IF SM-TEXT-LEN > 66
```

```
       THEN
         MOVE SPACES              TO SM-MSG-DATA-X
         MOVE SM-MSG-DATA-2       TO SM-MSG-DATA-X
         MOVE DISP-SERVER-MSG-5X TO RSLTNO(8)
         IF SM-TEXT-LEN > 132
           THEN
             MOVE SPACES             TO SM-MSG-DATA-X
             MOVE SM-MSG-DATA-3      TO SM-MSG-DATA-X
             MOVE DISP-SERVER-MSG-5X TO RSLTNO(9)
             IF SM-TEXT-LEN > 198
               THEN
                 MOVE SPACES             TO SM-MSG-DATA-X
                 MOVE SM-MSG-DATA-4      TO SM-MSG-DATA-X
                 MOVE DISP-SERVER-MSG-5X TO RSLTNO(10)
             END-IF
         END-IF
     END-IF.

 RETRIEVE-SERVER-MSGS-EXIT.
     EXIT.

*============================================================
*==                                                        ==
*== Subroutine to clear the output screen                  ==
*==                                                        ==
*============================================================
 CLEAR-SCREEN-DATA.

     MOVE SPACES TO RSLTNO( I2 ).

 CLEAR-SCREEN-DATA-EXIT.

     EXIT.

*============================================================
*==                                                        ==
*== Subroutine to handle MAPFAIL condition                 ==
*==                                                        ==
*============================================================
 NO-INPUT.
*---------

     MOVE 'Please Enter Input Fields' TO MSG-TEXT-1.

     GO TO GET-INPUT-AGAIN.
```

```
*============================================================
*==                                                        ==
*== Subroutine to handle AID condition                     ==
*==                                                        ==
*============================================================
 GETOUT.
*--------

     EXEC CICS RETURN END-EXEC.

     STOP RUN.

*============================================================
*==                                                        ==
*== Subroutine to handle ERROR condition                   ==
*==                                                        ==
*============================================================
 ERRORS.
*--------

     EXEC CICS DUMP DUMPCODE('ERRS') END-EXEC.

    STOP RUN.
```

APPENDIX B

Sample RPC Application

This appendix contains a sample Open ClientConnect application program, Sample program – SYCTSAR5, that sends an RPC to an Adaptive Server Enterprise or Open ServerConnect running in a CICS/IMS region.

The purpose of this sample program is to demonstrate the use of Client-Library functions, particularly those designed to send RPC requests. In some cases, one Client-Library function is used for demonstration purposes when another function would be more efficient. In order to best illustrate the flow of processing, the program does not do extensive error checking.

The remote procedure or transaction initiated by this RPC is called SYR2, which uses data from the sample table SYBASE.SAMPLETB.

- If your server is an Adaptive Server Enterprise, the remote procedure and table must be created by Transaction Router Service (TRS). A script is provided with TRS that it can use to create SYR2 and the sample table on Adaptive Server Enterprise.
- If your remote server is Open ServerConnect, SYR2 and SYBASE.SAMPLETB are provided on the product API tape.

SYCTSAR5 is provided as part of the Open ClientConnect package on the API tape.

Note The following additional sample programs are provided as part of the Open ClientConnect package on the API tape:

- SYCTSAD5, which demonstrates how to send an RPC request with parameters to an Adaptive Server Enterprise or Open ServerConnect running in a CICS region.
- SYCTSAV5, which demonstrates how to send an RPC request with parameters to an Adaptive Server Enterprise or Open ServerConnect running in a CICS/IMS region.

Sample program – SYCTSAR5

```
*@(#) syctsar5.cobol 11.2 12/14/95


 ****************************************************************
 *
 * Confidential property of Sybase, Inc.
 * (c) Copyright Sybase, Inc. 1985 TO 1997.
 * All rights reserved.
 *
 ****************************************************************
 ******* SYCTSAR5 - Client Language Request APPL - COBOL - CICS **
 **
 **  CICS TRANID:   SYR5
 **  PROGRAM:  SYCTSAR5
 **
 **  PURPOSE:  Demonstrates Open Client for CICS CALLs.
 **
 **  FUNCTION: Illustrates how to send an RPC request with
 **            parameters to:
 **
 **              - A SQL Server
 **              - An Open Server running in a CICS/IMS region.
 **
 **            SQL Server:
 **
 **              If the request is sent to a SQL Server it
 **              intitiates the stored procedure "SYR2".
 **
 **               Note: The Net-Gateway/MCG product includes a script
 **                     that creates this procedure in a target SQL
 **                     server.
 **
 **            Open Server/CICS or Open Server/IMS:
 **
 **              If the request is sent to an Open Server/CICS or
 **              IMS region, it initiates the transaction SYR2.
 **
 **              Note: Both Open Server/CICS and IMS products
 **                    include the sample transaction SYR2. This
 **                    is the server side transaction invoked by
 **                    this transaction.
 **
 **  PREREQS:  Before running SYCTSAR5, make sure that the server
 **            you wish to access has an entry in the Connection
```

```
**            Router Table for that Server and the MCG(s) that
**            you wish to use.
**
**
**  INPUT:    On the input screen, make sure to enter the Server
**            name, user id, and password for the target server.
**            TRAN NAME is not used for LAN servers.
**
**            If the target server is in a CICS or IMS region,
**            enter SYR2 in the TRAN NAME field.
**
**
**  Open Client CALLs used in this sample:
**
**    CSBCTXALLOC   allocate a context
**    CSBCTXDROP    drop a context
**    CTBBIND       bind a column variable
**    CTBCLOSE      close a server connection
**    CTBCMDALLOC   allocate a command
**    CTBCMDDROP    drop a command
**    CTBCOMMAND    initiate remote procedure CALL
**    CTBCONALLOC   allocate a connection
**    CTBCONDROP    drop a connection
**    CTBCONPROPS   alter properties of a connection
**    CTBCONNECT    open a server connection
**    CTBDESCRIBE   return a description of RESULT data
**    CTBDIAG       retrieve SQLCODE messages
**    CTBEXIT       exit client library
**    CTBFETCH      FETCH RESULT data
**    CTBINIT       init client library
**    CTBPARAM      define a command PARAMETER
**    CTBRESULTS    set up RESULT data
**    CTBSEND       send a request TO the server
**
** History:
**
** Date    BTS#   Descrition
** ======= ====== ============================================
** Feb1795        Create
** Oct1895 99999  Rewrite and add front end to the program
**
*****************************************************************

 IDENTIFICATION DIVISION.
 PROGRAM-ID. SYCTSAR5.
 ENVIRONMENT DIVISION.
```

```
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. XYZ.
 OBJECT-COMPUTER. XYZ.

 DATA DIVISION.
 WORKING-STORAGE SECTION.

*****************************************************************
** Client Library Cobol Copy Book
*****************************************************************

 COPY CTPUBLIC.

*****************************************************************
** CICS BMS DEFINITIONS
*****************************************************************

   COPY SYCTBA5.

*****************************************************************
*  Standard CICS Attribute and Print Control Chararcter List
*****************************************************************

   COPY DFHBMSCA.

*****************************************************************
** CICS Standard Attention Identifiers Cobol Copy Book
*****************************************************************

   COPY DFHAID.

*****************************************************************
*    CONSTANTS
*****************************************************************

 01  MSG-TEXT-1               PIC X(70) VALUE ' '.
 01  MSG-TEXT-2               PIC X(70)
                                VALUE 'Press Clear To Exit'.
 01  PAGE-CNT                 PIC S9(4) COMP VALUE +0.
 01  UTIME                    PIC S9(15) COMP-3.
 01  TMP-DATE                 PIC X(08).
 01  TMP-TIME                 PIC X(08).
 01  ENTER-DATA-SW              PIC X(01) VALUE 'N'.

 01  C-N                      PIC X(01) VALUE 'N'.
 01  C-Y                      PIC X(01) VALUE 'Y'.
```

```
01  MAX-SCREEN-ROWS           PIC S9(4) VALUE +10.

01 RESTYPE                    PIC S9(9) COMP SYNC VALUE IS 0.
01 NETDRIVER                  PIC S9(9) COMP SYNC VALUE IS 9999.
01 DATALEN                    PIC S9(9) COMP SYNC VALUE IS 0.
01 INTARG                     PIC S9(9) COMP SYNC VALUE IS 0.
01 INDIC                      PIC S9(9) COMP SYNC VALUE IS 0.
01 CMDSTR                     PIC X(200) VALUE IS SPACES.
01 STATUS-BIND                PIC S9(9) COMP SYNC VALUE IS 0.
01 STATUS-OK                  PIC S9(9) COMP SYNC VALUE IS 0.

01  BAD-INPUT                 PIC X(01) VALUE 'N'.

01  NO-MORE-MSGS-SW           PIC X(01).
    88  NO-MORE-MSGS            VALUE 'Y'.

01  NO-ERRORS-SW              PIC X(01).
    88  NO-ERRORS               VALUE 'N'.

01  SWITCHES.
    05 SW-RESULTS             PIC X(01) VALUE 'Y'.
       88  NO-MORE-RESULTS      VALUE 'N'.
    05 SW-FETCH               PIC X(01) VALUE 'Y'.
       88  NO-MORE-ROWS          VALUE 'N'.
    05 SW-DIAG                PIC X(01) VALUE 'N'.
       88  DIAG-MSGS-INITIALIZED VALUE 'Y'.

01  CS-LIB-MISC-FIELDS.
    05  CSL-CMD-HANDLE        PIC S9(9) COMP VALUE +0.
    05  CSL-CON-HANDLE        PIC S9(9) COMP VALUE +0.
    05  CSL-CTX-HANDLE        PIC S9(9) COMP VALUE +0.
    05  CSL-NULL              PIC S9(9) COMP VALUE +0.
    05  CSL-RC                PIC S9(9) COMP VALUE +0.

01  PROPS-FIELDS.
    05  PF-SERVER             PIC X(30) VALUE IS SPACES.
    05  PF-SERVER-SIZE        PIC S9(9) COMP VALUE +0.
    05  PF-USER               PIC X(08) VALUE IS SPACES.
    05  PF-USER-SIZE          PIC S9(9) COMP VALUE +0.
    05  PF-PWD                PIC X(08) VALUE IS SPACES.
    05  PF-PWD-SIZE           PIC S9(9) COMP VALUE +0.
    05  PF-TRAN               PIC X(08) VALUE IS SPACES.
    05  PF-TRAN-SIZE          PIC S9(9) COMP VALUE +0.
    05  PF-NETDRV             PIC X(08) VALUE IS SPACES.
    05  PF-DRV-SIZE           PIC S9(9) COMP VALUE +0.
    05  PF-DEPT               PIC X(03) VALUE 'D11'.
```

```
     05  PF-DEPT-SIZE          PIC S9(9) COMP VALUE +3.
     05  PF-STRLEN             PIC S9(9) COMP.
     05  PF-MSGLIMIT           PIC S9(9) COMP.

01 PARM1                       PIC S9(9) COMP SYNC.
01 PARM2.
   49 PLEN-RET                 PIC S9(4) COMP SYNC.
   49 PARR-RET                 PIC X(3) VALUE IS SPACES.

01 DISP-PARM.
   05 FILLER                   PIC X(1) VALUE IS '('.
   05 RETPARM-VAL              PIC 99999.
   05 RET-PARMMSG              PIC X(17) VALUE IS
                                      ' row(s) affected)'.
   05 FILLER                   PIC X(50) VALUE IS SPACES.

01 DISP-ROW.
   05 ROW1-VAL                 PIC X(12) VALUE IS SPACES.
   05  FILLER                  PIC X(01) VALUE IS SPACES.
   05 ROW2-VAL                 PIC X(15) VALUE IS SPACES.
   05  FILLER                  PIC X(01) VALUE IS SPACES.
   05 ROW3-VAL                 PIC zz9-.
   05  FILLER                  PIC X(08) VALUE IS SPACES.
   05 ROW4-VAL                 PIC zz.-.
   05  FILLER                  PIC X(06) VALUE IS SPACES.
   05 ROW5-VAL.
       49   LOW-VAL            PIC  ZZ,ZZZ.99-.
   05  FILLER                  PIC X(14) VALUE IS SPACES.

01 ROW1-BIND.
   49  ROW1-LEN                PIC S9(4) COMP SYNC VALUE IS 0.
   49  ROW1-TEXT               PIC X(12) VALUE IS SPACES.

01 ROW2-BIND.
   49  ROW2-LEN                PIC S9(4) COMP SYNC VALUE IS 0.
   49  ROW2-TEXT               PIC X(15) VALUE IS SPACES.

01 ROW3-BIND                   PIC S9(4) COMP SYNC VALUE IS 0.

01 ROW4-BIND.
   49 HIGH4-BIND               PIC S9(9) COMP SYNC VALUE IS 0.
   49 LOW4-BIND                PIC S9(5)V9(4) COMP SYNC
                                             VALUE IS 0.
01 ROW5-BIND.
   49 HIGH-BIND                PIC S9(9) COMP SYNC VALUE IS 0.
   49 LOW-BIND                 PIC S9(5)V9(4) COMP SYNC
```

```
                                                VALUE IS 0.
01 OUTLEN                      PIC S9(9) COMP SYNC VALUE IS 0.
01 NUMROWS                     PIC S9(9) COMP SYNC VALUE IS 0.
01 I                           PIC S9(9) COMP SYNC VALUE IS 0.
01 I1                          PIC S9(9) COMP SYNC VALUE IS 0.
01 I2                          PIC S9(9) COMP SYNC VALUE IS 0.
01 COPIED                      PIC S9(9) COMP SYNC VALUE IS 0.
01 COPIED-NULL                 PIC S9(9) COMP SYNC VALUE IS 0.
01 INDICATOR                   PIC S9(9) COMP SYNC VALUE IS 0.
01 INDICATOR-NULL              PIC S9(9) COMP SYNC VALUE IS 0.

01 DISP-MSG.
   05 TEST-CASE                PIC X(08) VALUE IS 'SYCTSAR5'.
   05 FILLER                   PIC X(01) VALUE IS SPACES.
   05 MSG.
      10 SAMP-LIT              PIC X(05) VALUE IS 'rc = '.
      10 SAMP-RC               PIC -Z9.
      10 FILLER                PIC X(02) VALUE IS ', '.
      10 REST-LIT              PIC X(12) VALUE IS
                                  'Result Type:'.
      10 REST-TYPE             PIC Z(3)9.
      10 FILLER                PIC X(03) VALUE IS SPACES.
      10 MSGSTR                PIC X(40) VALUE IS SPACES.

01 DATAFMT-PARM.
       05 NM-PARM              PIC X(132).
       05 NMLEN-PARM           PIC S9(9) COMP SYNC.
       05 DATATYPE-PARM        PIC S9(9) COMP SYNC.
       05 FORMT-PARM           PIC S9(9) COMP SYNC.
       05 MAXLENGTH-PARM       PIC S9(9) COMP SYNC.
       05 SCALE-PARM           PIC S9(9) COMP SYNC.
       05 PRECISION-PARM       PIC S9(9) COMP SYNC.
       05 FMTSTATUS-PARM       PIC S9(9) COMP SYNC.
       05 FMTCOUNT-PARM        PIC S9(9) COMP SYNC.
       05 USERTYPE-PARM        PIC S9(9) COMP SYNC.
       05 LOCALE-PARM          PIC X(68).

01 DATAFMT-BIND.
       05 NM-BIND              PIC X(132).
       05 NMLEN-BIND           PIC S9(9) COMP SYNC.
       05 DATATYPE-BIND        PIC S9(9) COMP SYNC.
       05 FORMT-BIND           PIC S9(9) COMP SYNC.
       05 MAXLENGTH-BIND       PIC S9(9) COMP SYNC.
       05 SCALE-BIND           PIC S9(9) COMP SYNC.
       05 PRECISION-BIND       PIC S9(9) COMP SYNC.
       05 FMTSTATUS-BIND       PIC S9(9) COMP SYNC.
```

```
        05 FMTCOUNT-BIND       PIC S9(9) COMP SYNC.
        05 USERTYPE-BIND       PIC S9(9) COMP SYNC.
        05 LOCALE-BIND         PIC X(68).

 01 WCOLUMN                    PIC S9(9) COMP SYNC.

 01  DIAG-FIELDS.
     05  DF-MSGNO              PIC S9(9) COMP VALUE +1.
     05  DF-NUM-OF-MSGS        PIC S9(9) COMP VALUE +0.

*******************************
** Client Message Structure  **
*******************************

 01  CLIENT-MSG.
     05  CM-SEVERITY           PIC S9(9) COMP SYNC.
     05  CM-MSGNO              PIC S9(9) COMP SYNC.
     05  CM-TEXT               PIC X(256).
     05  CM-TEXT-LEN           PIC S9(9) COMP SYNC.
     05  CM-OS-MSGNO           PIC S9(9) COMP SYNC.
     05  CM-OS-MSGTXT          PIC X(256).
     05  CM-OS-MSGTEXT-LEN     PIC S9(9) COMP SYNC.
     05  CM-STATUS             PIC S9(9) COMP.

 01  DISP-CLIENT-MSG-HDR.
     05  CLIENT-MSG-HDR        PIC X(15) VALUE IS
                                   'Client Message:'.

 01  DISP-CLIENT-MSG-1.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  CM-SEVERITY-HDR       PIC X(09) VALUE IS 'Severity:'.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  CM-SEVERITY-DATA      PIC Z(8)9.
     05  CM-STATUS-HDR         PIC X(12) VALUE IS
                                   ',  Status:  '.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  CM-STATUS-DATA        PIC Z(8)9.

 01  DISP-CLIENT-MSG-2.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  CM-OC-MSGNO-HDR       PIC X(09) VALUE IS 'OC MsgNo:'.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  CM-OC-MSGNO-DATA      PIC Z(8)9.

 01  DISP-CLIENT-MSG-3.
     05  FILLER                PIC X(02) VALUE IS SPACES.
```

```
      05  CM-OC-MSG-HDR          PIC X(09) VALUE IS 'OC MsgTx:'.
      05  FILLER                 PIC X(02) VALUE IS SPACES.
      05  CM-OC-MSG-DATA         PIC X(66).

  01  DISP-CLIENT-MSG-3A.
      05  CM-OC-MSG-DATA-1       PIC X(66).
      05  CM-OC-MSG-DATA-2       PIC X(66).
      05  CM-OC-MSG-DATA-3       PIC X(66).
      05  CM-OC-MSG-DATA-4       PIC X(58).

  01  DISP-CLIENT-MSG-3B.
      05  FILLER                 PIC X(13) VALUE IS SPACES.
      05  CM-OC-MSG-DATA-X       PIC X(66).

  01  DISP-EMPTY-CLIENT-MSG-3.
      05  FILLER                 PIC X(02) VALUE IS SPACES.
      05  CM-OC-MSG-HDR          PIC X(09) VALUE IS 'OC MsgTx:'.
      05  FILLER                 PIC X(02) VALUE IS SPACES.
      05  NO-DATA                PIC X(11) VALUE IS 'No Message!'.

  01  DISP-CLIENT-MSG-4.
      05  FILLER                 PIC X(02) VALUE IS SPACES.
      05  CM-OS-MSG-HDR          PIC X(09) VALUE IS 'OS MsgNo:'.
      05  FILLER                 PIC X(02) VALUE IS SPACES.
      05  CM-OS-MSGNO-DATA       PIC Z(8)9.

  01  DISP-CLIENT-MSG-5.
      05  FILLER                 PIC X(02) VALUE IS SPACES.
      05  CM-OS-MSG-HDR          PIC X(09) VALUE IS 'OS MsgTx:'.
      05  FILLER                 PIC X(02) VALUE IS SPACES.
      05  CM-OS-MSG-DATA         PIC X(66).

  01  DISP-CLIENT-MSG-5A.
      05  CM-OS-MSG-DATA-1       PIC X(66).
      05  CM-OS-MSG-DATA-2       PIC X(66).
      05  CM-OS-MSG-DATA-3       PIC X(66).
      05  CM-OS-MSG-DATA-4       PIC X(58).

  01  DISP-EMPTY-CLIENT-MSG-5.
      05  FILLER                 PIC X(02) VALUE IS SPACES.
      05  CM-OS-MSG-HDR          PIC X(09) VALUE IS 'OS MsgTx:'.
      05  FILLER                 PIC X(02) VALUE IS SPACES.
      05  NO-DATA                PIC X(11) VALUE IS 'No Message!'.

 ******************************
 ** Server Message Structure  **
```

```
*******************************

 01  SERVER-MSG.
     05  SM-MSGNO              PIC S9(9) COMP.
     05  SM-STATE              PIC S9(9) COMP.
     05  SM-SEV                PIC S9(9) COMP.
     05  SM-TEXT               PIC X(256).
     05  SM-TEXT-LEN           PIC S9(9) COMP.
     05  SM-SVRNAME            PIC X(256).
     05  SM-SVRNAME-LEN        PIC S9(9) COMP.
     05  SM-PROC               PIC X(256).
     05  SM-PROC-LEN           PIC S9(9) COMP.
     05  SM-LINE               PIC S9(9) COMP.
     05  SM-STATUS             PIC S9(9) COMP.

 01  DISP-SERVER-MSG-HDR.
     05  SERVER-MSG-HDR        PIC X(15) VALUE IS
                                   'Server Message:'.

 01  DISP-SERVER-MSG-1.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  SM-MSG-NO-HDR         PIC X(09) VALUE IS
                                   'Message#:'.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  SM-MSG-NO-DATA        PIC Z(8)9.
     05  SM-SEVERITY-HDR       PIC X(12) VALUE IS
                                   ',  Severity:'.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  SM-SEVERITY-DATA      PIC Z(8)9.
     05  SM-STATE-HDR          PIC X(12) VALUE IS
                                   ',  State No:'.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  SM-STATE-DATA         PIC Z(8)9.

 01  DISP-SERVER-MSG-2.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  SM-LINE-NO-HDR        PIC X(09) VALUE IS
                                   'Line  No:'.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  SM-LINE-NO-DATA       PIC Z(8)9.
     05  SM-STATUS-HDR         PIC X(12) VALUE IS
                                   ',  Status  :'.
     05  FILLER                PIC X(02) VALUE IS SPACES.
     05  SM-STATUS-DATA        PIC Z(8)9.

 01  DISP-SERVER-MSG-3.
```

```
     05  FILLER                 PIC X(02) VALUE IS SPACES.
     05  SM-SVRNAME-HDR         PIC X(09) VALUE IS 'Serv Nam:'.
     05  FILLER                 PIC X(02) VALUE IS SPACES.
     05  SM-SVRNAME-DATA        PIC X(66).
     05  FILLER                 PIC X(03) VALUE IS '...'.

 01  DISP-SERVER-MSG-4.
     05  FILLER                 PIC X(02) VALUE IS SPACES.
     05  SM-PROC-ID-HDR         PIC X(09) VALUE IS 'Proc  ID:'.
     05  FILLER                 PIC X(02) VALUE IS SPACES.
     05  SM-PROC-ID-DATA        PIC X(66).

 01  DISP-SERVER-MSG-5.
     05  FILLER                 PIC X(02) VALUE IS SPACES.
     05  SM-MSG-HDR             PIC X(09) VALUE IS 'Message :'.
     05  FILLER                 PIC X(02) VALUE IS SPACES.
     05  SM-MSG-DATA            PIC X(66).

 01  DISP-SERVER-MSG-5A.
     05  SM-MSG-DATA-1          PIC X(66).
     05  SM-MSG-DATA-2          PIC X(66).
     05  SM-MSG-DATA-3          PIC X(66).
     05  SM-MSG-DATA-4          PIC X(58).

 01  DISP-SERVER-MSG-5X.
     05  FILLER                 PIC X(13) VALUE IS SPACES.
     05  SM-MSG-DATA-X          PIC X(66).

 01  CICS-FIELDS.
     05  CICS-RESPONSE          PIC S9(9) COMP.

 01  QUERY-FIELDS.
     05  QF-LEN                 PIC S9(4) COMP VALUE +1.
     05  QF-MAXLEN              PIC S9(4) COMP VALUE +1.
     05  QF-ANSWER              PIC X(01) VALUE IS SPACES.

*****************************************************************
 PROCEDURE DIVISION.
*****************************************************************

*************************
* CICS Condition Handler *
*************************

     EXEC CICS HANDLE CONDITION MAPFAIL(NO-INPUT)
                                ERROR(ERRORS)
```

```
     END-EXEC.

 ********************
 * CICS Aid Handler *
 ********************

     EXEC CICS HANDLE AID ANYKEY(NO-INPUT)
                          CLEAR(GETOUT)
     END-EXEC.

 **************************
 * PROGRAM INITIALIZATION *
 **************************

     MOVE SPACES     TO DISP-ROW.
     MOVE C-N        TO NO-MORE-MSGS-SW.
     MOVE C-N        TO NO-ERRORS-SW.
     MOVE C-Y        TO SW-DIAG.

     MOVE LOW-VALUES TO A5PANELO.
     MOVE -1         TO SERVERL.

     COMPUTE PAGE-CNT = PAGE-CNT + 1.

     PERFORM GET-SYSTEM-TIME.

  GET-INPUT-AGAIN.

     PERFORM DISPLAY-INITIAL-SCREEN.

     PERFORM GET-INPUT-DATA.

 *******************************
 * ALLOCATE A CONTEXT STRUCTURE *
 *******************************

     MOVE ZERO       TO CSL-CTX-HANDLE.
     MOVE LOW-VALUES TO DATAFMT-PARM DATAFMT-BIND DISP-ROW.

     CALL 'CSBCTXAL' USING CS-VERSION-50
                           CSL-RC
                           CSL-CTX-HANDLE.

      IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
          MOVE SPACES TO MSGSTR
```

```
           STRING 'CSBCTXAL failed'
                  DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
       END-IF.

 *********************************
 * INTITIALIZE THE CLIENT-LIBRARY *
 *********************************

      CALL 'CTBINIT' USING CSL-CTX-HANDLE
                           CSL-RC
                           CS-VERSION-50.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBINIT failed'
                  DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
      END-IF.

      PERFORM PROCESS-INPUT.

      PERFORM QUIT-CLIENT-LIBRARY.

      GOBACK.



*============================================================
 *==                                                        ==
 *== Subroutine to get system date/time                     ==
 *==                                                        ==
 *============================================================
  GET-SYSTEM-TIME.
 *----------------

      EXEC CICS ASKTIME
                ABSTIME(UTIME)
      END-EXEC.

      EXEC CICS FORMATTIME
                ABSTIME(UTIME)
                DATESEP('/')
```

```
                MMDDYY(TMP-DATE)
                TIME(TMP-TIME)
                TIMESEP
     END-EXEC.

*=========================================================
*==                                                     ==
*== Subroutine to display SYD5 initial screen           ==
*==                                                     ==
*=========================================================
 DISPLAY-INITIAL-SCREEN.
*-----------------------

     MOVE TMP-DATE   TO SDATEO.
     MOVE TMP-TIME   TO STIMEO.
     MOVE 'SYCTSAR5' TO PROGNMO.

     MOVE PAGE-CNT   TO SPAGEO.
     MOVE MSG-TEXT-1 TO MSG1O.
     MOVE MSG-TEXT-2 TO MSG2O.

     EXEC CICS SEND MAP('A5PANEL')
                    MAPSET('SYCTBA5')
                    CURSOR
                    FRSET
                    ERASE
                    FREEKB
     END-EXEC.

*=========================================================
*==                                                     ==
*== Subroutine to get input data                        ==
*==                                                     ==
*=========================================================
 GET-INPUT-DATA.
*---------------

     EXEC CICS RECEIVE MAP('A5PANEL')
                    MAPSET('SYCTBA5')
                    ASIS
     END-EXEC.

     IF SERVERL = ZERO
       THEN
         IF PF-SERVER = SPACES
           THEN
```

```
  MOVE 'Please Enter Server Name' TO MSG-TEXT-1
       MOVE -1                          TO SERVERL
       MOVE C-Y                         TO ENTER-DATA-SW
    END-IF
  ELSE
    MOVE SERVERI   TO PF-SERVER
    MOVE SERVERL   TO PF-SERVER-SIZE
END-IF.

IF USERL = ZERO
  THEN
    IF PF-USER = SPACES
      THEN
        MOVE 'Please Enter User-ID' TO MSG-TEXT-1
        MOVE -1                     TO USERL
        MOVE C-Y                    TO ENTER-DATA-SW
    END-IF
  ELSE
    MOVE USERI   TO PF-USER
    MOVE USERL   TO PF-USER-SIZE
    MOVE PF-USER TO USERO
END-IF.

IF PSWDL NOT EQUAL ZERO
  THEN
    MOVE PSWDI TO PF-PWD
    MOVE PSWDL TO PF-PWD-SIZE
END-IF

IF TRANL NOT EQUAL ZERO
  THEN
    MOVE TRANI TO PF-TRAN
    MOVE TRANL TO PF-TRAN-SIZE
END-IF.

IF NETDRVL NOT EQUAL ZERO
  THEN
    MOVE NETDRVI TO PF-NETDRV
    MOVE NETDRVL TO PF-DRV-SIZE
END-IF.

IF ENTER-DATA-SW = C-Y
  THEN
    MOVE C-N TO ENTER-DATA-SW
    PERFORM DISPLAY-INITIAL-SCREEN
    PERFORM GET-INPUT-DATA
```

```
     END-IF.

*=========================================================
*==                                                     ==
*== Subroutine to process input data                    ==
*==                                                     ==
*=========================================================
 PROCESS-INPUT.

*********************************
* ALLOCATE A CONNECTION HANDLE *
*********************************

     MOVE ZERO TO CSL-CON-HANDLE.

     CALL 'CTBCONAL' USING CSL-CTX-HANDLE
                           CSL-RC
                           CSL-CON-HANDLE.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBCONAL failed'
                 DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
     END-IF.

******************
* SET THE USER ID *
******************

     CALL 'CTBCONPR' USING CSL-CON-HANDLE
                           CSL-RC
                           CS-SET
                           CS-USERNAME
                           PF-USER
                           PF-USER-SIZE
                           CS-FALSE
                           OUTLEN.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBCONPR for user-id failed'
                 DELIMITED BY SIZE INTO MSGSTR
```

```
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
         END-IF.

********************
* SET THE PASSWORD *
********************

      CALL 'CTBCONPR' USING CSL-CON-HANDLE
                            CSL-RC
                            CS-SET
                            CS-PASSWORD
                            PF-PWD
                            PF-PWD-SIZE
                            CS-FALSE
                            OUTLEN.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBCONPR for password failed'
                  DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
        END-IF.


********************
* SET THE TRAN NAME *
********************

      IF PF-TRAN-SIZE IS NOT EQUAL TO ZEROES THEN

          CALL 'CTBCONPR' USING CSL-CON-HANDLE
                                CSL-RC
                                CS-SET
                                CS-TRANSACTION-NAME
                                PF-TRAN
                                PF-TRAN-SIZE
                                CS-FALSE
                                OUTLEN

          IF CSL-RC NOT EQUAL CS-SUCCEED
            THEN
               MOVE SPACES TO MSGSTR
               STRING 'CTBCONPR for TRAN name failed'
```

```
                    DELIMITED BY SIZE INTO MSGSTR
               PERFORM PRINT-MSG
               PERFORM ALL-DONE
             END-IF

      END-IF.



*******************************
 * SET THE NET DRIVER PROPERTY *
 *******************************

      IF PF-NETDRV = SPACES OR PF-NETDRV = 'LU62'                  X
                           OR PF-NETDRV = 'lu62'
          MOVE CS-LU62 TO NETDRIVER
      ELSE
        IF PF-NETDRV = 'IBMTCPIP' OR PF-NETDRV = 'ibmtcpip'
          MOVE CS-TCPIP TO NETDRIVER
      ELSE
        IF PF-NETDRV = 'INTERLIN' OR PF-NETDRV = 'interlin'
          MOVE CS-INTERLINK TO NETDRIVER
      ELSE
        IF PF-NETDRV = 'CPIC' OR PF-NETDRV = 'cpic'
          MOVE CS-NCPIC TO NETDRIVER
      END-IF.

      IF PF-DRV-SIZE IS NOT EQUAL TO ZEROES THEN

          CALL 'CTBCONPR' USING CSL-CON-HANDLE
                                CSL-RC
                                CS-SET
                                CS-NET-DRIVER
                                NETDRIVER
                                CS-UNUSED
                                CS-FALSE
                                OUTLEN

          IF CSL-RC NOT EQUAL CS-SUCCEED
            THEN
               MOVE SPACES TO MSGSTR
               STRING 'CTBCONPR for network driver failed'
                      DELIMITED BY SIZE INTO MSGSTR
               PERFORM PRINT-MSG
               PERFORM ALL-DONE
            END-IF
```

```
     END-IF.

*************************
* CONNECT TO THE SERVER *
*************************

     CALL 'CTBCONNE' USING CSL-CON-HANDLE
                           CSL-RC
                           PF-SERVER
                           PF-SERVER-SIZE
                           CS-FALSE.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBCONNE failed' DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
       END-IF.

     IF NO-ERRORS
       THEN
         PERFORM SEND-PARAM THRU SEND-PARAM-EXIT
     END-IF.

************************************
* PROCESS THE RESULTS OF THE COMMAND *
************************************

     IF NO-ERRORS
       THEN
         PERFORM RESULTS-PROCESSING UNTIL NO-MORE-RESULTS
         PERFORM CLOSE-CONNECTION
     END-IF.

 PROCESS-INPUT-EXIT.
     EXIT.

*============================================================
*==                                                        ==
*== Subroutine to allocate, send, and process commands     ==
*==                                                        ==
*============================================================
 SEND-PARAM.
```

```
 ****************************
 * NOW GET A COMMAND HANDLE. *
 ****************************

      MOVE ZERO TO CSL-CMD-HANDLE.

      CALL 'CTBCMDAL' USING CSL-CON-HANDLE
                            CSL-RC
                            CSL-CMD-HANDLE.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBCMDAL failed'
                 DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.



**************************************************************
 * INITIATE THE STORED PROCEDURE "SYR2". THE DATA WILL BE   *
 * RETURNED FROM THE TABLE SYBASE.SAMPLETB. THIS CAN EITHER *
 * BE A DB2 OR A SQL SERVER TABLE DEPENDING  ON WHETHER     *
 * THE RPC IS SENT TO A CICS REGION OR A SQL SERVER.        *
 **************************************************************

      MOVE LOW-VALUES TO CMDSTR.
      MOVE 4          TO INTARG.
      STRING 'SYR2' DELIMITED BY SIZE INTO CMDSTR.

      CALL 'CTBCOMMA' USING CSL-CMD-HANDLE
                            CSL-RC
                            CS-RPC-CMD
                            CMDSTR
                            INTARG
                            CS-UNUSED.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBCOMMAND failed'
                 DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
```

```
     END-IF.

*****************************
* SET UP THE RPC PARAMETERS *
*****************************

     MOVE '@parm1'         TO NM-PARM.
     MOVE 6                TO NMLEN-PARM.
     MOVE CS-FMT-NULLTERM TO FORMT-PARM.
     MOVE CS-RETURN        TO FMTSTATUS-PARM.
     MOVE CS-INT-TYPE      TO DATATYPE-PARM.
     MOVE LENGTH OF PARM1 TO DATALEN.
     MOVE 0                TO PARM1.

     CALL 'CTBPARAM' USING CSL-CMD-HANDLE
                           CSL-RC
                           DATAFMT-PARM
                           PARM1
                           DATALEN
                           INDIC.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBPARAM CS-INT-TYPE parm1 failed'
                DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
     END-IF.

     MOVE '@parm2'            TO NM-PARM.
     MOVE 6                   TO NMLEN-PARM.
     MOVE CS-FMT-NULLTERM     TO FORMT-PARM.
     MOVE CS-INPUTVALUE       TO FMTSTATUS-PARM.
     MOVE CS-VARCHAR-TYPE     TO DATATYPE-PARM.
     MOVE PF-DEPT             TO PARR-RET.
     MOVE PF-DEPT-SIZE        TO DATALEN.
     MOVE 255                 TO MAXLENGTH-PARM.

     CALL 'CTBPARAM' USING CSL-CMD-HANDLE
                           CSL-RC
                           DATAFMT-PARM
                           PARM2
                           DATALEN
                           INDIC.
```

```
      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBPARAM CS-VARCHAR-TYPE parm2 failed'
                 DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.

 ************************************
 * SEND THE COMMAND AND THE PARAMETERS *
 ************************************

      CALL 'CTBSEND' USING CSL-CMD-HANDLE
                           CSL-RC.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBSEND failed'
                 DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.

  SEND-PARAM-EXIT.
      EXIT.

 *=======================================================
 *==                                                   ==
 *== Subroutine to process result                      ==
 *==                                                   ==
 *=======================================================
  RESULTS-PROCESSING.

 ***************************
 * SET UP THE RESULTS DATA *
 ***************************

      CALL 'CTBRESUL' USING CSL-CMD-HANDLE
                            CSL-RC
                            RESTYPE.

 ************************************************
 * DETERMINE THE OUTCOME OF THE COMMAND EXECUTION *
 ************************************************
```

```
     EVALUATE CSL-RC

         WHEN CS-SUCCEED

*************************************************************
* DETERMINE THE TYPE OF RESULT RETURNED BY THE CURRENT REQUEST *
*************************************************************

             EVALUATE RESTYPE

**********************
* PROCESS ROW RESULTS *
**********************

               WHEN CS-ROW-RESULT
                 PERFORM RESULT-ROW-PROCESSING
                 MOVE 'Y' TO SW-FETCH
                 PERFORM FETCH-ROW-PROCESSING UNTIL NO-MORE-ROWS

*************************************************************
* PROCESS PARAMETER RESULTS - THERE SHOULD BE NO PARAMETERS *
* TO PROCESS                                                *
*************************************************************

               WHEN CS-PARAM-RESULT
                 PERFORM RESULT-PARAM-PROCESSING
                 MOVE 'Y' TO SW-FETCH
                 PERFORM FETCH-PARAM-PROCESSING

*****************************************************
* PROCESS STATUS RESULTS - THE STORED PROCEDURE RESULT *
* WILL BE PROCESSED IN THIS EXAMPLE                 *
*****************************************************

               WHEN CS-STATUS-RESULT
                 MOVE 'Y' TO SW-FETCH
                 CALL 'CTBFETCH' USING CSL-CMD-HANDLE
                                       CSL-RC
                                       CS-UNUSED
                                       CS-UNUSED
                                       CS-UNUSED
                                       NUMROWS

                 IF CSL-RC = CS-FAIL
                   THEN
```

```
                    MOVE SPACES TO MSGSTR
                    STRING 'CTBFETCH status failed'
                         DELIMITED BY SIZE INTO MSGSTR
                    PERFORM PRINT-MSG
                    PERFORM ALL-DONE
                  END-IF

 ************************************************************
 * PRINT AN ERROR MESSAGE IF THE SERVER ENCOUNTERED AN ERROR *
 * WHILE EXECUTING THE REQUEST                               *
 ************************************************************

               WHEN CS-CMD-FAIL
                 STRING
                   'CTBRESUL failed with CS-CMD-FAIL restype'
                         DELIMITED BY SIZE INTO MSGSTR
                 PERFORM PRINT-MSG

 ****************************************************************
 * PRINT A MESSSAGE FOR SUCCESSFUL COMMANDS THAT RETURNED NO DATA *
 * (OPTIONAL)                                                     *
 ****************************************************************

               WHEN CS-CMD-SUCCEED
                 STRING
                   'CTBRESUL returned CS-CMD-SUCCEED restype'
                         DELIMITED BY SIZE INTO MSGSTR



*********************************************************
 * PRINT A MESSSAGE FOR REQUESTS THAT HAVE BEEN PROCESSED *
 * SUCCESSFULLY (OPTIONAL)                                *
 *********************************************************

               WHEN CS-CMD-DONE
                 STRING
                   'CTBRESUL returned CS-CMD-DONE restype'
                         DELIMITED BY SIZE INTO MSGSTR

               WHEN OTHER
                 STRING
                   'CTBRESUL returned UNKNOW restype'
                         DELIMITED BY SIZE INTO MSGSTR
                  PERFORM PRINT-MSG
                  MOVE 'N' TO SW-RESULTS
```

```
             END-EVALUATE

*******************************************************
* PRINT AN ERROR MESSAGE IF THE CTBRESULTS CALL FAILED *
*******************************************************

          WHEN CS-FAIL
            MOVE 'N' TO SW-RESULTS
            STRING
              'CTBRESUL failed with CS-FAIL ret-cd'
                  DELIMITED BY SIZE INTO MSGSTR
            PERFORM PRINT-MSG

************************************************************
* DROP OUT OF THE RESULTS LOOP IF NO MORE RESULT SETS ARE   *
* AVAILABLE FOR PROCESSING OR IF THE RESULTS WERE CANCELLED *
************************************************************

          WHEN CS-END-RESULTS
            MOVE 'N' TO SW-RESULTS

          WHEN CS-CANCELLED
            MOVE 'N' TO SW-RESULTS

          WHEN OTHER
            MOVE 'N' TO SW-RESULTS
            STRING 'CTBRESUL failed with UNKNOWN ret-cd'
                  DELIMITED BY SIZE INTO MSGSTR
            PERFORM PRINT-MSG

      END-EVALUATE.

      MOVE 0 TO RESTYPE.


*============================================================
*==                                                        ==
*== Subroutine to process result rows                      ==
*==                                                        ==
*============================================================
 RESULT-ROW-PROCESSING.
*======================

*********************************
* FOR EACH COLUMN BIND THE RESULT *
```

```
 ***********************************

      PERFORM BIND-ROW-PROCESSING.
      MOVE 1 TO I2.
      STRING
        'FirstName    LastName         EducLvl    JobCode   Salary'
        DELIMITED BY SIZE INTO RSLTNO(I2).
      MOVE 2 TO I2.
      STRING '===========  ===============  ======='
             DELIMITED BY SIZE
             '    =======   ==========='
             DELIMITED BY SIZE
             INTO RSLTNO(I2).

 *=====================================================
 *==                                                 ==
 *== Subroutine to describe the returned parameters ==
 *==                                                 ==
 *=====================================================
  RESULT-PARAM-PROCESSING.

 ********************************************
 * RETURN A DESCRIPTION OF THE RETURN PARAMETER *
 ********************************************

      MOVE 1 TO I.
      CALL 'CTBDESCR' USING CSL-CMD-HANDLE
                            CSL-RC
                            I
                            DATAFMT-BIND.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBDESCR failed'
                 DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.



****************************
 * BIND THE RETURN PARAMETER *
 ****************************
```

```
     PERFORM BIND-PARAM-PROCESSING.

*=========================================================
*==                                                     ==
*== Subroutine to fetch row processing                  ==
*==                                                     ==
*=========================================================
 FETCH-ROW-PROCESSING.

******************
* FETCH THE ROWS *
******************

     CALL 'CTBFETCH' USING CSL-CMD-HANDLE
                           CSL-RC
                           CS-UNUSED
                           CS-UNUSED
                           CS-UNUSED
                           NUMROWS.

     EVALUATE CSL-RC

*********************************************
* MOVE THE ROW DATA TO PRINTABLE DATA FORMATS *
*********************************************

         WHEN CS-SUCCEED
              COMPUTE I2 EQUAL I2 + 1
              MOVE 'Y'       TO SW-FETCH
              MOVE LOW-BIND  TO LOW-VAL
              MOVE ROW3-BIND TO ROW3-VAL
              MOVE LOW4-BIND TO ROW4-VAL
              MOVE ROW1-TEXT TO ROW1-VAL
              MOVE ROW2-TEXT TO ROW2-VAL

              IF I2 > MAX-SCREEN-ROWS
                THEN
                  MOVE SPACES TO MSG-TEXT-2
                  STRING 'Please press return to continue!'
                           DELIMITED BY SIZE INTO MSG1O
                  PERFORM DISP-DATA
                  PERFORM CLEAR-SCREEN-DATA
                          VARYING I2 FROM 1 BY 1
                          UNTIL I2 > MAX-SCREEN-ROWS
                  COMPUTE PAGE-CNT = PAGE-CNT + 1
                  MOVE 1 TO I2
```

```
                STRING
                   'FirstName    LastName         EducLvl'
                   DELIMITED BY SIZE
                   '    JobCode   Salary'
                   DELIMITED BY SIZE
                   INTO RSLTNO(I2)
                MOVE 2 TO I2
                STRING
                   '===========  ===============  ========'
                   DELIMITED BY SIZE
                   '    =======   ==========='
                   DELIMITED BY SIZE
                   INTO RSLTNO(I2)
                MOVE 3 TO I2
             END-IF
             MOVE DISP-ROW TO RSLTNO (I2)
             MOVE SPACES   TO ROW1-TEXT ROW2-TEXT
             MOVE SPACES   TO ROW1-VAL ROW2-VAL



*************************************************************
 * PRINT THE ROWS AFTER ALL ROW DATA HAS BEEN FETCHED *
 *************************************************************

          WHEN CS-END-DATA
               MOVE 'Press Clear To Exit'
                        TO MSG-TEXT-2
               MOVE 'N' TO SW-FETCH
               STRING 'All rows processing completed!'
                      DELIMITED BY SIZE INTO MSG1O
               PERFORM DISP-DATA

 ************************************************************
 * DROP OUT OF THE FETCH LOOP IF THE CTBFETCH COMMAND FAILS *
 ************************************************************

          WHEN CS-FAIL
               MOVE 'N' TO SW-FETCH
               STRING 'CTBFETCH returned CS-FAIL ret-cd'
                      DELIMITED BY SIZE INTO MSGSTR
               PERFORM PRINT-MSG

 ************************************************************
 * DROP OUT OF THE FETCH LOOP IF A RECOVERABLE COMMAND FAILS *
 * WHILE FETCHING A ROW OR IF THE OPERATION WAS CANCELLED    *
```

```
************************************************************

         WHEN CS-ROW-FAIL
              MOVE 'N' TO SW-FETCH
              STRING 'CTBFETCH returned CS-ROW-FETCH ret-cd'
                     DELIMITED BY SIZE INTO MSGSTR
              PERFORM PRINT-MSG

         WHEN CS-CANCELLED
              MOVE 'N' TO SW-FETCH
              STRING 'CTBFETCH returned CS-CANCELLED ret-cd'
                     DELIMITED BY SIZE INTO MSGSTR
              PERFORM PRINT-MSG

         WHEN OTHER
              MOVE 'N' TO SW-FETCH
              STRING 'CTBFETCH returned UNKNOWN ret-cd'
                     DELIMITED BY SIZE INTO MSGSTR
              PERFORM PRINT-MSG

     END-EVALUATE.

*=========================================================
*==                                                     ==
*== Subroutine to fetch return parameter                ==
*==                                                     ==
*=========================================================
 FETCH-PARAM-PROCESSING.
*=======================

*****************************
* FETCH THE RETURN PARAMETER *
*****************************

     CALL 'CTBFETCH' USING CSL-CMD-HANDLE
                           CSL-RC
                           CS-UNUSED
                           CS-UNUSED
                           CS-UNUSED
                           NUMROWS.

     IF CSL-RC = CS-FAIL
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBFETCH return parameter failed'
                         DELIMITED BY SIZE INTO MSGSTR
```

```
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.

 ****************************************************************
 * MOVE THE PARAMETER DATA TO A PRINTABLE DATA FORMAT AND PRINT *
 * THE DATA                                                     *
 ****************************************************************

      COMPUTE I2 EQUAL I2 + 1.
      MOVE PARM1     TO RETPARM-VAL.
      MOVE DISP-PARM TO RSLTNO (I2).



*=========================================================
 *==                                                     ==
 *== Subroutine to bind row processing                   ==
 *==                                                     ==
 *=========================================================
  BIND-ROW-PROCESSING.

 ******************************************************
 * BIND THE COLUMNS RETURNED FROM THE STORED PROCEDURE *
 ******************************************************

        MOVE 1                TO WCOLUMN.
        MOVE CS-VARCHAR-TYPE  TO DATATYPE-BIND.
        MOVE CS-MAX-CHAR      TO MAXLENGTH-BIND.
        MOVE CS-FMT-NULLTERM  TO FORMT-BIND.
        MOVE CS-PARAM-NOTNULL TO INDICATOR-NULL.
        MOVE CS-PARAM-NOTNULL TO COPIED-NULL.

        CALL 'CTBBIND' USING CSL-CMD-HANDLE
                             CSL-RC
                             WCOLUMN
                             DATAFMT-BIND
                             ROW1-BIND
                             COPIED
                             COPIED-NULL
                             INDICATOR
                             INDICATOR-NULL .

        IF CSL-RC NOT EQUAL CS-SUCCEED
          THEN
            MOVE SPACES TO MSGSTR
```

```
    STRING 'CTBBIND CS-VARCHAR-TYPE column 1 failed'
           DELIMITED BY SIZE INTO MSGSTR
    PERFORM PRINT-MSG
    PERFORM ALL-DONE
END-IF.

MOVE 2 TO WCOLUMN.

CALL 'CTBBIND' USING CSL-CMD-HANDLE
                     CSL-RC
                     WCOLUMN
                     DATAFMT-BIND
                     ROW2-BIND
                     COPIED
                     COPIED-NULL
                     INDICATOR
                     INDICATOR-NULL .

IF CSL-RC NOT EQUAL CS-SUCCEED
  THEN
    MOVE SPACES TO MSGSTR
    STRING 'CTBBIND CS-VARCHAR-TYPE column 2 failed'
           DELIMITED BY SIZE INTO MSGSTR
    PERFORM PRINT-MSG
    PERFORM ALL-DONE
END-IF.

MOVE 3                    TO WCOLUMN.
MOVE CS-SMALLINT-TYPE     TO DATATYPE-BIND.
MOVE LENGTH OF ROW3-BIND TO MAXLENGTH-BIND.
MOVE CS-FMT-UNUSED        TO FORMT-BIND.

CALL 'CTBBIND' USING CSL-CMD-HANDLE
                     CSL-RC
                     WCOLUMN
                     DATAFMT-BIND
                     ROW3-BIND
                     COPIED
                     COPIED-NULL
                     INDICATOR
                     INDICATOR-NULL .

IF CSL-RC NOT EQUAL CS-SUCCEED
  THEN
    MOVE SPACES TO MSGSTR
    STRING 'CTBBIND CS-SMALLINT-TYPE column 3 failed'
```

```
           DELIMITED BY SIZE INTO MSGSTR
    PERFORM PRINT-MSG
    PERFORM ALL-DONE
END-IF.

MOVE 4                    TO WCOLUMN.
MOVE LENGTH OF ROW4-BIND TO MAXLENGTH-BIND.
MOVE CS-MONEY-TYPE        TO DATATYPE-BIND.
MOVE CS-FMT-UNUSED        TO FORMT-BIND.
MOVE CS-SRC-VALUE         TO PRECISION-BIND.
MOVE CS-SRC-VALUE         TO SCALE-BIND.

CALL 'CTBBIND' USING CSL-CMD-HANDLE
                     CSL-RC
                     WCOLUMN
                     DATAFMT-BIND
                     ROW4-BIND
                     COPIED
                     COPIED-NULL
                     INDICATOR
                     INDICATOR-NULL .

IF CSL-RC NOT EQUAL CS-SUCCEED
  THEN
    MOVE SPACES TO MSGSTR
    STRING 'CTBBIND CS-MONEY-TYPE column 4 failed'
           DELIMITED BY SIZE INTO MSGSTR
    PERFORM PRINT-MSG
    PERFORM ALL-DONE
END-IF.

MOVE 5                    TO WCOLUMN.
MOVE LENGTH OF ROW5-BIND TO MAXLENGTH-BIND.

CALL 'CTBBIND' USING CSL-CMD-HANDLE
                     CSL-RC
                     WCOLUMN
                     DATAFMT-BIND
                     ROW5-BIND
                     COPIED
                     COPIED-NULL
                     INDICATOR
                     INDICATOR-NULL .

IF CSL-RC NOT EQUAL CS-SUCCEED
  THEN
```

```
           MOVE SPACES TO MSGSTR
           STRING 'CTBBIND CS-MONEY-TYPE column 5 failed'
                  DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
       END-IF.

*=========================================================
*==                                                     ==
*== Subroutine to bind return parameters               ==
*==                                                     ==
*=========================================================
 BIND-PARAM-PROCESSING.

*****************************
* BIND THE RETURN PARAMETER *
*****************************

       MOVE 1           TO WCOLUMN.
       MOVE CS-INT-TYPE TO DATATYPE-BIND.

       CALL 'CTBBIND' USING CSL-CMD-HANDLE
                            CSL-RC
                            WCOLUMN
                            DATAFMT-BIND
                            PARM1
                            COPIED
                            COPIED-NULL
                            INDICATOR
                            INDICATOR-NULL .

       IF CSL-RC NOT EQUAL CS-SUCCEED
         THEN
           MOVE SPACES TO MSGSTR
           STRING 'CTBBIND for return parameter failed'
                  DELIMITED BY SIZE INTO MSGSTR
           PERFORM PRINT-MSG
           PERFORM ALL-DONE
       END-IF.

*=========================================================
*==                                                     ==
*== Subroutine to display output                       ==
*==                                                     ==
*=========================================================
 DISP-DATA.
```

```
      MOVE TMP-DATE   TO SDATEO.
      MOVE TMP-TIME   TO STIMEO.
      MOVE 'SYCTSAR5' TO PROGNMO.
      MOVE PAGE-CNT   TO SPAGEO.

      MOVE DFHBMPRO   TO SERVERA.
      MOVE PF-SERVER  TO SERVERO.

      MOVE DFHBMPRO   TO USERA.
      MOVE PF-USER    TO USERO.

      MOVE DFHBMPRO   TO NETDRVA.
      MOVE PF-NETDRV  TO NETDRVO.

      MOVE DFHBMDAR   TO PSWDA.
      MOVE PF-PWD    TO PSWDO.
      MOVE MSG-TEXT-2 TO MSG2O.

********************
* DISPLAY THE DATA *
********************

      EXEC CICS SEND MAP('A5PANEL')
                     MAPSET('SYCTBA5')
                     CURSOR
                     FRSET
                     ERASE
                     FREEKB
      END-EXEC.

      EXEC CICS RECEIVE INTO(QF-ANSWER)
                        LENGTH(QF-LEN)
                        MAXLENGTH(QF-MAXLEN)
                        RESP(CICS-RESPONSE)
      END-EXEC.

 DISP-DATA-EXIT.
      EXIT.
 *=========================================================
*==                                                       ==
*== Subroutine to print output messages.                  ==
*==                                                       ==
*=========================================================
 PRINT-MSG.
```

```
      MOVE CSL-RC  TO SAMP-RC.
      MOVE RESTYPE TO REST-TYPE.

      IF DIAG-MSGS-INITIALIZED AND BAD-INPUT EQUAL TO C-N
        THEN
          PERFORM GET-DIAG-MESSAGES
      END-IF.

 ***********************
 * DISPLAY THE MESSAGE *
 ***********************

      IF NO-ERRORS
        THEN
          PERFORM DISP-DATA
      END-IF.

      MOVE C-Y TO NO-ERRORS-SW.
      MOVE SPACES TO MSGSTR.
      MOVE ZERO   TO SAMP-RC.
      MOVE ZERO   TO REST-TYPE.

  PRINT-MSG-EXIT.
      EXIT.

 *============================================================
 *==                                                        ==
 *== Subroutine to drop and to deallocate all handlers,     ==
 *== to close server connection and exit client library     ==
 *==                                                        ==
 *============================================================
  ALL-DONE.

      PERFORM CLOSE-CONNECTION.
      PERFORM QUIT-CLIENT-LIBRARY.
      STOP RUN.

  ALL-DONE-EXIT.
      EXIT.



*============================================================
 *==                                                      ==
 *== Subroutine to perform drop command handler, close    ==
 *== server connection, and deallocate Connection Handler. ==
```

```
 *==                                                          ==
 *============================================================
  CLOSE-CONNECTION.

 ***************************
 * DROP THE COMMAND HANDLE *
 ***************************

      CALL 'CTBCMDDR' USING CSL-CMD-HANDLE
                            CSL-RC.

      IF CSL-RC = CS-FAIL
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBCMDDR failed'
                 DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
      END-IF.



*******************************
 * CLOSE THE SERVER CONNECTION *
 *******************************

      CALL 'CTBCLOSE' USING CSL-CON-HANDLE
                            CSL-RC
                            CS-UNUSED.

      IF CSL-RC = CS-FAIL
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBCLOSE failed'
                 DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
      END-IF.

 ************************************
 * DE-ALLOCATE THE CONNECTION HANDLE *
 ************************************

      CALL 'CTBCONDR' USING CSL-CON-HANDLE
                            CSL-RC.

      IF CSL-RC = CS-FAIL
        THEN
```

```
         MOVE SPACES TO MSGSTR
         STRING 'CTBCONDR failed'
                DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
     END-IF.

 CLOSE-CONNECTION-EXIT.
     EXIT.

*=================================================
*==                                             ==
*== Subroutine to perform exit client library and ==
*== deallocate context structure.                ==
*==                                             ==
*=================================================
 QUIT-CLIENT-LIBRARY.

**************************
* EXIT THE CLIENT LIBRARY *
**************************

     CALL 'CTBEXIT' USING CSL-CTX-HANDLE
                          CSL-RC
                          CS-UNUSED.
     IF CSL-RC = CS-FAIL
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBEXIT failed'
                DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
     END-IF.

***********************************
* DE-ALLOCATE THE CONTEXT STRUCTURE *
***********************************

     CALL 'CSBCTXDR' USING CSL-CTX-HANDLE
                           CSL-RC.

     IF CSL-RC = CS-FAIL
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CSBCTXDR failed'
                DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
     END-IF.
```

```
 QUIT-CLIENT-LIBRARY-EXIT.
     EXIT.

*=================================================
*==                                             ==
*== Subroutine to retrieve any diagnostic messages ==
*==                                             ==
*=================================================
 GET-DIAG-MESSAGES.

***********************************
* Disable calls to this subroutine *
***********************************

     MOVE 'N' TO SW-DIAG.

*****************************
* First, get client messages *
*****************************

     CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                          CSL-RC,
                          CS-UNUSED,
                          CS-STATUS,
                          CS-CLIENTMSG-TYPE,
                          CS-UNUSED,
                          DF-NUM-OF-MSGS.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         MOVE SPACES TO MSGSTR
         STRING 'CTBDIAG CS-STATUS CS-CLIENTMSG-TYPE failed'
                DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
       ELSE
         IF DF-NUM-OF-MSGS > 0
           THEN
             PERFORM RETRIEVE-CLIENT-MSGS
                 VARYING I FROM 1 BY 1
                     UNTIL I IS GREATER THAN DF-NUM-OF-MSGS
         END-IF
     END-IF.

****************************
```

```
* Then, get server messages *
*****************************

     CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                          CSL-RC,
                          CS-UNUSED,
                          CS-STATUS,
                          CS-SERVERMSG-TYPE,
                          CS-UNUSED,
                          DF-NUM-OF-MSGS.

     IF CSL-RC NOT EQUAL CS-SUCCEED
       THEN
         STRING 'CTBDIAG CS-STATUS CS-SERVERMSG-TYPE failed'
                DELIMITED BY SIZE INTO MSGSTR
         PERFORM PRINT-MSG
         PERFORM ALL-DONE
       ELSE
         IF DF-NUM-OF-MSGS > 0
           THEN
             PERFORM RETRIEVE-SERVER-MSGS
                 VARYING I FROM 1 BY 1
                     UNTIL I IS GREATER THAN DF-NUM-OF-MSGS
         END-IF
     END-IF.

 GET-DIAG-MESSAGES-EXIT.
     EXIT.

*============================================================
*==                                                        ==
*== Subroutine to retrieve diagnostic messages from client ==
*==                                                        ==
*============================================================
 RETRIEVE-CLIENT-MSGS.

     MOVE 1 TO I1.

     CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                          CSL-RC,
                          CS-UNUSED,
                          CS-GET,
                          CS-CLIENTMSG-TYPE,
                          DF-MSGNO,
                          CLIENT-MSG.
```

```
      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBDIAG CS-GET CS-CLIENTMSG-TYPE failed'
                 DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.

 ************************
 * display message text *
 ************************

      MOVE DISP-CLIENT-MSG-HDR TO RSLTNO( I1 ).
      MOVE 3                   TO I1.

      MOVE CM-SEVERITY         TO CM-SEVERITY-DATA.
      MOVE CM-STATUS           TO CM-STATUS-DATA.
      MOVE DISP-CLIENT-MSG-1   TO RSLTNO( I1 ).
      COMPUTE I1 EQUAL I1 + 1

      MOVE CM-MSGNO            TO CM-OC-MSGNO-DATA.
      MOVE DISP-CLIENT-MSG-2   TO RSLTNO( I1 ).
      COMPUTE I1 EQUAL I1 + 1

      IF CM-MSGNO NOT EQUAL 0
        THEN
          MOVE SPACES            TO CM-OC-MSG-DATA
          MOVE CM-TEXT           TO CM-OC-MSG-DATA
          MOVE CM-TEXT           TO DISP-CLIENT-MSG-3A
          MOVE DISP-CLIENT-MSG-3 TO RSLTNO( I1 )
          COMPUTE I1 EQUAL I1 + 1
          IF CM-TEXT-LEN > 66
            THEN
              MOVE CM-OC-MSG-DATA-2   TO CM-OC-MSG-DATA-X
              MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
              COMPUTE I1 EQUAL I1 + 1
              IF CM-TEXT-LEN > 132
                THEN
                  MOVE SPACES             TO CM-OC-MSG-DATA-X
                  MOVE CM-OC-MSG-DATA-3   TO CM-OC-MSG-DATA-X
                  MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
                  COMPUTE I1 EQUAL I1 + 1
                  IF CM-TEXT-LEN > 198
                    THEN
                      MOVE SPACES             TO CM-OC-MSG-DATA-X
```

```
                MOVE CM-OC-MSG-DATA-4   TO CM-OC-MSG-DATA-X
                MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
                COMPUTE I1 EQUAL I1 + 1
              END-IF
            END-IF
          END-IF
      ELSE
        MOVE DISP-EMPTY-CLIENT-MSG-3 TO RSLTNO( I1 )
        COMPUTE I1 EQUAL I1 + 1
    END-IF.

    MOVE CM-OS-MSGNO       TO CM-OS-MSGNO-DATA.
    MOVE DISP-CLIENT-MSG-4 TO RSLTNO( I1 ).
    COMPUTE I1 EQUAL I1 + 1

    IF CM-OS-MSGNO NOT EQUAL 0
      THEN
        MOVE SPACES            TO CM-OS-MSG-DATA
        MOVE CM-OS-MSGTXT      TO CM-OS-MSG-DATA
        MOVE SPACES            TO DISP-CLIENT-MSG-5A
        MOVE CM-OS-MSGTXT      TO DISP-CLIENT-MSG-5A
        MOVE DISP-CLIENT-MSG-5 TO RSLTNO( I1 )
        COMPUTE I1 EQUAL I1 + 1
        IF CM-OS-MSGTEXT-LEN > 66
          THEN
            MOVE SPACES             TO CM-OC-MSG-DATA-X
            MOVE CM-OS-MSG-DATA-2   TO CM-OC-MSG-DATA-X
            MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
            COMPUTE I1 EQUAL I1 + 1
            IF CM-OS-MSGTEXT-LEN > 132
              THEN
                MOVE SPACES             TO CM-OC-MSG-DATA-X
                MOVE CM-OS-MSG-DATA-3   TO CM-OC-MSG-DATA-X
                MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
                COMPUTE I1 EQUAL I1 + 1
                IF CM-OS-MSGTEXT-LEN > 198
                  THEN
                    MOVE SPACES             TO CM-OC-MSG-DATA-X
                    MOVE CM-OS-MSG-DATA-4   TO CM-OC-MSG-DATA-X
                    MOVE DISP-CLIENT-MSG-3B TO RSLTNO( I1 )
                    COMPUTE I1 EQUAL I1 + 1
                END-IF
            END-IF
        END-IF
      ELSE
        MOVE DISP-EMPTY-CLIENT-MSG-5 TO RSLTNO( I1 )
```

```
          COMPUTE I1 EQUAL I1 + 1
      END-IF.

 RETRIEVE-CLIENT-MSGS-EXIT.
      EXIT.

*============================================================
*==                                                        ==
*== Subroutine to retrieve diagnostic messages from server ==
*==                                                        ==
*============================================================
 RETRIEVE-SERVER-MSGS.

      CALL 'CTBDIAG' USING CSL-CON-HANDLE,
                           CSL-RC,
                           CS-UNUSED,
                           CS-GET,
                           CS-SERVERMSG-TYPE,
                           DF-MSGNO,
                           SERVER-MSG.

      IF CSL-RC NOT EQUAL CS-SUCCEED
        THEN
          MOVE SPACES TO MSGSTR
          STRING 'CTBDIAG CS-GET CS-SERVERMSG-TYPE failed'
                         DELIMITED BY SIZE INTO MSGSTR
          PERFORM PRINT-MSG
          PERFORM ALL-DONE
      END-IF.

************************
* display message text *
************************

      MOVE SM-MSGNO   TO SM-MSG-NO-DATA.
      MOVE SM-SEV     TO SM-SEVERITY-DATA.
      MOVE SM-STATE   TO SM-STATE-DATA.

      MOVE SM-LINE    TO SM-LINE-NO-DATA.
      MOVE SM-STATUS  TO SM-STATUS-DATA.

      MOVE SPACES     TO SM-SVRNAME-DATA.
      MOVE SM-SVRNAME TO SM-SVRNAME-DATA.

      MOVE SPACES     TO SM-PROC-ID-DATA.
      MOVE SM-PROC    TO SM-PROC-ID-DATA.
```

```
      MOVE SPACES     TO SM-MSG-DATA.
      MOVE SM-TEXT    TO SM-MSG-DATA.

      MOVE SPACES     TO DISP-SERVER-MSG-5A.
      MOVE SM-TEXT    TO DISP-SERVER-MSG-5A.


      MOVE DISP-SERVER-MSG-HDR TO RSLTNO (1).
      MOVE DISP-SERVER-MSG-1   TO RSLTNO (3).
      MOVE DISP-SERVER-MSG-2   TO RSLTNO (4).
      MOVE DISP-SERVER-MSG-3   TO RSLTNO (5).
      MOVE DISP-SERVER-MSG-4   TO RSLTNO (6).

      MOVE DISP-SERVER-MSG-5   TO RSLTNO (7).
      IF SM-TEXT-LEN > 66
        THEN
          MOVE SPACES             TO SM-MSG-DATA-X
          MOVE SM-MSG-DATA-2      TO SM-MSG-DATA-X
          MOVE DISP-SERVER-MSG-5X TO RSLTNO(8)
          IF SM-TEXT-LEN > 132
            THEN
              MOVE SPACES             TO SM-MSG-DATA-X
              MOVE SM-MSG-DATA-3      TO SM-MSG-DATA-X
              MOVE DISP-SERVER-MSG-5X TO RSLTNO(9)
              IF SM-TEXT-LEN > 198
                THEN
                  MOVE SPACES             TO SM-MSG-DATA-X
                  MOVE SM-MSG-DATA-4      TO SM-MSG-DATA-X
                  MOVE DISP-SERVER-MSG-5X TO RSLTNO(10)
              END-IF
          END-IF
      END-IF.

  RETRIEVE-SERVER-MSGS-EXIT.
      EXIT.



*============================================================
 *==                                                        ==
 *== Subroutine to clear the output screen                  ==
 *==                                                        ==
 *============================================================
  CLEAR-SCREEN-DATA.
```

```
      MOVE SPACES TO RSLTNO( I2 ).

  CLEAR-SCREEN-DATA-EXIT.

      EXIT.

 *============================================================
 *==                                                        ==
 *== Subroutine to handle MAPFAIL condition                 ==
 *==                                                        ==
 *============================================================
  NO-INPUT.
 *---------

      MOVE 'Please Enter Input Fields' TO MSG-TEXT-1.

      GO TO GET-INPUT-AGAIN.



*============================================================
 *==                                                        ==
 *== Subroutine to handle AID condition                     ==
 *==                                                        ==
 *============================================================
  GETOUT.
 *--------

      EXEC CICS RETURN END-EXEC.

      STOP RUN.

 *============================================================
 *==                                                        ==
 *== Subroutine to handle ERROR condition                   ==
 *==                                                        ==
 *============================================================
  ERRORS.
 *--------

      EXEC CICS DUMP DUMPCODE('ERRS') END-EXEC.

      STOP RUN.
```

APPENDIX C

Sybase Product Documentation

This appendix summarizes Mainframe Connect documentation by content and by audience.

This appendix includes the following topics:

- Publications by content
- Publications by audience

Note For instructions on ordering documentation, go to the Sybase web site at http://www.sybase.com.

Publications by content

Table C-1 includes a synopsis of each publication in the current documentation set.

Table C-1: Documentation description

Title	Contents
Mainframe Connect Client Option and Server Option *Messages and Codes*	Provides details on messages that Mainframe Connect components return.
Mainframe Connect Server Option for CICS *Installation and Administration Guide*	Describes configuring the Mainframe Connect network, installing Open ServerConnect, setting up security, and troubleshooting for an MVS-CICS environment.
Mainframe Connect Server Option for IMS and MVS *Installation and Administration Guide*	Describes configuring the Mainframe Connect network, setting up APPC communications, installing Open ServerConnect, setting up security, and troubleshooting for an IMS or MVS environment.

Title	Contents
Mainframe Connect Server Option *Programmer's Reference for C*	Provides reference material for writing Open ServerConnect programs that call C Gateway-Library functions. This guide contains reference pages for Gateway-Library routines and descriptions of the underlying concepts for C programmers.
Mainframe Connect Server Option *Programmer's Reference for PL/1*	Provides reference material for writing Open ServerConnect programs that call PL/I Gateway-Library functions. This guide contains reference pages for Gateway-Library routines and descriptions of the underlying concepts for PL/I programmers.
Mainframe Connect Server Option *Programmer's Reference for RSPs*	Provides information for anyone who designs, codes, and tests remote stored procedures (RSPs).
Mainframe Connect Client Option for CICS *Installation and Administration Guide*	Describes installing and configuring Open ClientConnect, routing requests to a server, and using Sybase isql. This manual also contains instructions for using the connection router and the mainframe-based ISQL utility.
Mainframe Connect Client Option for IMS and MVS *Installation and Administration Guide*	Describes installing Open ClientConnect, routing requests to a server, and using Sybase isql. This manual also contains instructions for using mainframe-based ISQL utility.
Mainframe Connect Client Option *Programmer's Reference for PL/1*	Describes writing Open ClientConnect programs that call PL/I Client-Library functions. This guide contains reference pages for Client-Library routines and descriptions of the underlying concepts for PL/I programmers.
Mainframe Connect Client Option *Programmer's Reference for C*	Describes writing Open ClientConnect programs that call C Client-Library functions. This guide contains reference pages for Client-Library routines and descriptions of the underlying concepts for C programmers.
Mainframe Connect Client Option *Programmer's Reference for CSAs*	Provides information for anyone who designs, codes, and tests client services applications (CSAs).
Mainframe Connect DB2 UDB Option for CICS *Installation and Administration Guide*	Describes configuring the mainframe, installing MainframeConnect, setting up security, and troubleshooting for a CICS environment.
Enterprise Connect Data Access and Mainframe Connect *Server Administration Guide* for DirectConnect	Describes administration of the DirectConnect server. Information about administering specific service libraries and services is provided in other DirectConnect publications.
Mainframe Connect DirectConnect for z/OS Option *Installation Guide*	Describes installing a DirectConnect server and service libraries.

Title	Contents
Mainframe Connect DirectConnect for z/OS Option *User's Guide for Transaction Router Services*	Describes configuring, controlling, and monitoring DirectConnect Transaction Router Service Library, as well as setting up security.
Mainframe Connect DirectConnect for z/OS Option *User's Guide for DB2 Access Services* (for use with MainframeConnect for DB2 UDB)	Describes configuring, controlling, and monitoring DirectConnect for OS/390 Access Service, as well as setting up security.

Publications by audience

Table C-2 on page 376 lists the publications in the documentation set and shows the intended audience for each book. The symbols used in the table are:

- R = required for this role
- O = optional (can be useful for this role)

Table C-2: Documentation by audience

Title	Mainframe systems support	Mainframe application developer	Direct Connect & Net-Gateway 3.0.x Admin.	Workstation application developer
Mainframe Connect Client Option and Server Option *Messages and Codes*	R	R	R	R
Mainframe Connect Server Option for CICS *Installation and Administration Guide*	R		O	
Mainframe Connect Server Option for IMS and MVS *Installation and Administration Guide*	R		O	
Mainframe Connect Server Option *Programmer's Reference for C*	R	R		
Mainframe Connect Server Option *Programmer's Reference for PL/1*	R	R		
Mainframe Connect Server Option *Programmer's Reference for RSPs*	R	R		
Mainframe Connect Client Option for CICS *Installation and Administration Guide*	R		O	
Mainframe Connect Client Option for IMS and MVS *Installation and Administration Guide*	R		O	
Mainframe Connect Client Option *Programmer's Reference for COBOL*	R	R		
Mainframe Connect Client Option *Programmer's Reference for PL/1*	R	R		
Mainframe Connect Client Option *Programmer's Reference for C*	R	R		
Mainframe Connect Client Option *Programmer's Reference for CSAs*	R	R		
Enterprise Connect Data Access and Mainframe Connect *Server Administration Guide* for DirectConnect	O		R	

Title	Mainframe systems support	Mainframe application developer	Direct Connect & Net-Gateway 3.0.x Admin.	Workstation application developer
Mainframe Connect DirectConnect for z/OS Option *Installation Guide*	O		R	
Mainframe Connect DirectConnect for z/OS Option *User's Guide for Transaction Router Services*	O		R	R
Mainframe Connect DirectConnect for z/OS Option *User's Guide for DB2 Access Services*	O		R	R

Index

A

Adaptive Server Enterprise
  message handling 177
APPC
  discussion of 11
Application name
  property 182, 185
Arguments
  assigning NULL to 180

B

Blanks
  stripping trailing blanks 166
BUFFER-LEN
  what to do when too short 166

C

Cancel
  results 35
Character datatypes
  list of 175
Character Set Conversion
  property 185
Character set conversion
  property 182
Choosing
  dynamic network drivers 9, 12, 13
  network drivers 9, 12, 13
CICS and LU6.2 11
CICS operating environment 12
Client message structure
  definition 168, 169
  severity values 168
Client messages
  description 177
Client-Library
  datatypes 173
  determining version 159
  error handling 177
  functions 14, 15, 18
  functions in mixed-mode program 15
  initializing 117
  programs 15, 18, 19, 21
  properties 180
  setting up environment 18
  version 117
CLIENTMSG structure
  data definition 168
  description 168, 169
  severity values 168
  used with CTBDIAG 87
Closing
  server connections 37
Columns
  binding result columns to program variables 25
Command handles
  allocating 41, 54
  assigning NULL to 180
  deallocating 44
  definition 201
  Gateway-Library equivalent 44
  retrieving properties 47
  routines that affect 201
  setting properties 47
Command parameters, defining 120
Command structures
  allocating 18
  deallocating 18, 21
  definition 16
Commands
  language commands 54
  processing results of 18
  sending 18, 142
  steps in sending to a server 53, 147
Communication block

property 182
Communications sessions block
property 185
Compatibility, Sybase mainframe access components 14
Connection handles
allocating 54, 55
assigning NULL to 180
deallocating 61
definition 200
Gateway-Library equivalent 41
properties 201
retrieving properties 70
routines that affect 201
setting properties 70
Connection Router
function of 20
Connection Router table 8
Connection structures
allocating 19
deallocating 18, 21
description 19
Connections
closing 18, 21, 37
establishing 18
forcing a close 40
max number of connections property 183
opening 20
setting maximum number of 186
Context handles
allocating 159
assigning NULL to 180
deallocating 162
definition 199
properties 199
retrieving properties 64
routines that affect 201
setting properties 64
Context structures
allocating 19
deallocating 18, 21
definition 15
description 19
Control structures. See Structures, control 15
Conversion
character set property 182
Criteria for choosing a network driver 12
cs_ctx_alloc
description 19
used in a program 18, 19
cs_ctx_drop
used in a program 18, 21
CS-APPNAME
description 182, 185
CSBCONFIG
description 148, 153
CSBCONVERT
conversions performed by 159
DATAFMT structure description 170
description 153
CSBCTXALLOC
description 159
CSBCTXDROP
description 162, 164
CS-CHARSETCNV
description 182
CS-CLEAR
when ACTION is CS-CLEAR 167
CS-COMMBLOCK
description 182
CS-EXTRA-INF
description 182
CS-GET
when ACTION is CS-GET 167
CS-HOSTNAME
description 182, 186
CS-LOC-PROP
description 182, 186
CS-LOGIN-STATUS
description 186
CS-LOGIN-TIMEOUT
description 183
CS-MAX-CONNECT
description 183, 186
CS-NET-DRIVER
description 183
CS-NETIO
description 183, 187
CS-NO-COUNT
meaning with CTBRESINFO 136
CS-NOINTERRUPT
description 183, 187
CS-PACKED370

FMT-SCALE used with 172
CS-PACKETSIZE
description 184, 187
CS-PASSWORD
description 184, 187
CS-SET
when ACTION is CS-SET 167
CS-TDS-VERSION
description 184, 187
CS-TEXTLIMIT
description 184, 188
CS-TIMEOUT
description 184, 188
CS-TRANSACTION-NAME
description 184, 188
CS-USERDATA
description 184, 189
CS-USERNAME
description 184, 189
CS-VERSION
description 189
ct_bind
used in a program 18, 20, 21
ct_close
used in a program 18, 21
ct_cmd_alloc
used in a program 18, 20
ct_cmd_drop
used in a program 18, 21
ct_command
used in a program 18, 20
ct_con_alloc
used in a program 18, 19
ct_con_drop
used in a program 18, 21
ct_con_props
description 19
used in a program 18, 19
ct_connect
used in a program 18, 20
ct_exit
used in a program 18, 21
ct_fetch
used in a program 18, 20
ct_init
used in a program 18, 19
ct_param
used in a program 21
ct_res_info
used in a program 18, 20
ct_results
loop 20
used in a program 18, 20
ct_send
used in a program 18, 20
CTBBIND
DATAFMT structure description 170
description 25, 34
CTBCANCEL
description 35, 37
CTBCLOSE
description 37, 40
CTBCMDALLOC
description 41, 43
CTBCMDDROP
description 44, 46
CTBCMDPROPS
description 47, 50
CTBCOMMAND
description 51, 54
CTBCONALLOC
description 54, 61
CTBCONDROP
description 63
CTBCONFIG
description 64, 67
max number of connections 186
CTBCONNECT
description 67, 70
determining TDS version 188
legal values for TDS version 188
CTBCONPROPS
description 77
CTBDESCRIBE
DATAFMT structure description 170
description 78, 85
CTBDIAG
CS-EXTRA-INF property 185
description 86, 104
CTBEXIT
description 104, 106
CTBFETCH

description 107, 112
CTBGETFORMAT
description 112, 115
CTBINIT
description 117, 119
CTBPARAM
DATAFMT structure description 170
description 120
CTBREMOTEPWD
description 127, 131
CTBRESINFO
description 131, 137
CTBRESULTS
description 137, 142
CTBSEND
description 142, 147
Customization
client password access code 169
national language 170
truncating LONG VARCHAR strings 170

D

Data descriptions
in DATAFMT structure 170
DATAFMT structure
data definition 170
description 170
fields in 171, 172
functions used by 172
Datatypes
character 175
converted by CSBCONVERT 159
converted by CTBBIND 33
correspondences 173, 177
data declarations for 173, 177
datetime 176
DB2 LONG VARCHAR truncation 170
decimal 176
discussion 173, 177
float 176
integer 176
list of supported 173, 177
money 177
numeric 176
real 176
specifying precision for packed decimal 173
specifying scale for packed decimal 172
Datetime datatypes
list of 176
Decimal datatype
description 176
Defining
command parameters 120
dynamic network drivers 9
network drivers 9
Dynamic network driver 19
choosing 9, 12, 13
criteria for choosing 12
defining 9
invoking 9
loading 9
network type and environment 12
operating environment 12
dynamic network driver
property 183

E

EIB
pointer to 182, 185
Environment
gateway-enabled 3, 4
gateway-less 3, 7
three-tier 3, 4
two-tier 3, 7
Error handling
description 177, 179
managing in-line error handling 86, 87
using SQLCA and SQLCODE 178
with CTBDIAG 86, 178
Error messages
client 177
how client messages are processed 168
how handled 177
server 177
severity level in client messages 168
Errors
SQLCODE used with 197
Extra information

property 182

F

Fetching
result columns 107
result data 107
return parameters 107
return status 107
Float datatype
description 176
FMT-FORMAT
supported values 172
FMT-MAXLEN
meaning with CSBCONVERT 172
meaning with CTBBIND 172
meaning with CTBDESCRIBE 172
FMT-PRECIS
when used 173
FMT-SCALE
when used 172
FMT-STATUS
legal values for 173
symbolic values of 173

G

Gateway-enabled
discussion of 3, 4
Gateway-less
discussion of 3, 7
Gateway-Library 15
functions in mixed-mode program 15

H

Handles
command 44, 47, 54
connection 55, 61, 70
context 64, 159
Host name
property 185

I

IHANDLE
Client-Library equivalent 55, 64
Image data
maximum length of image data property 184
Integer datatypes
list of 176
Interrupt indicator
property 183
Invoking
dynamic network drivers 9
network drivers 9
isql utility 8

L

Language commands
defining parameters for 125
Language request
initiating 50
Loading
dynamic network drivers 9
network drivers 9
Locale
property 182
Locale information
property 186
Login name
server login property 184
Login properties 181
Login status
property 186
Login timeout
property 183, 186
LOGOUT
closing the connection 37
LU6.2 11

M

Macros
SYGWDRIV 10
Mainframe Connect documentation
by audience 376

MCC
See Mainframe Client Connect 179
Messages
discussion 177, 179
how client messages are processed 168
severity level in client messages 168
SQLCA used with 196
types of with CTBDIAG 87
Mixed-mode
definition 15
where documented 15
Money datatypes
list of 177

N

Name
server login property 184
National languages
set during customization 170
Negotiated properties 181
Network communication definitions
choosing a network driver 10
overview 10
Network driver
choosing 9, 12, 13
criteria for choosing 12
defining 9
invoking 9
loading 9
network type and environment 12
operating environment 12
Network I/O
property 187
Network type and environment
dynamic network driver 12
network driver 12
No interrupt
property 187
Nulls
definition 180
discussion 180

O

Open ClientConnect
communication 3, 8
communication at the mainframe 8
communication at the server 8
determining version of 189
network configuration 3
security 9
Operating environment
CICS 12
dynamic network driver 12
network driver 12
OUTLEN
using to determine buffer length 166

P

Packed decimal
specifying precision 173
specifying scale 172
Packet size
property 184
Packets
packet size property 187
Parameter conventions 168
Parameter results
binding to program variables 25
Parameters
data descriptions in DATAFMT structure 170
defining 120
fetching 107
processing parameter results 20
return 191
unused parameters 180
Password
access code required for client 169
server password property 184
Passwords
password property 187
Precision
of datatype CS-PACKED370 173
specifying for packed decimal data 173
Programs
basic steps in 18
finishing up 18

mixed-mode 15
setting up the environment 18
writing 18
Properties
application name 182
character set conversion 182, 185
command handle properties 47
communication block 182
communications sessions block 185
connection 19
CS-APPNAME 185
defined at context level 19
discussion 180, 189
dynamic network driver 183
extra information 179, 182
host name 182, 185
interrupt indicator 183
locale information 186
localization 182
login properties 181
login status 186
login timeout 183, 186
maximum length of image data 184
maximum length of text data 184
maximum number of connections 183, 186
negotiated properties 181
network I/O 187
no interrupt 187
Open ClientConnect 189
packet size 184, 187
password 184, 187
server login name 184
server name 182
setting and retrieving properties 181
summary of properties table 181
synchronous I/O 183
TDS version 187
text and image limit 188
timeout 184
transaction name 184, 188
user data 189
user name 189
user-allocated data 184

R

Real datatype
description 176
Requests
initiating 50
Result set
definition 193
result_type
used in a program 21
Results
binding to program variables 25
cancelling 35, 141
command number 132, 134
data descriptions in DATAFMT structure 170
description 192
determining when completely processed 141
number of items 135
number of items returned 132
number of rows affected 132, 135
processing 20, 21
processing results 193
result types 138
run-time errors 142
setting up 137, 142
types of 138, 193
Return status
fetching 107
Return status results
binding to program variables 25
Returning
command handle information 47
Rows
processing result rows 20
RPCs
defining parameters for 126
discussion 189, 192
initiating 50
results 191
routines used with 190
server-to-server 190

S

SAA
discussion of 10

Scale
of datatypes 172
specifying for packed decimal data 172
Security 9
Server message structure
definition 194
severity values 195
Server messages
description 177
Server name
property 182
Server-Host Mapping table
about 8
SERVERMSG structure
data definition 194
description 196
severity values 195
used with CTBDIAG 87
Servers
closing a server connection 37
Severity level
in CLIENTMSG values 168
in SERVERMSG values 195
SNA
discussion of 11
SQLCA structure 196
description 196, 197
SQLCODE structure
description 197
Status
processing result status 21
return 192
Stored procedures
return parameters 191
return status 192
two ways to execute 190
Stripping blanks
using BUFBLANKSTRIP 166
Structures
commands 16
connection 19
context 15, 19
control 15, 16
discussion 199
SQLCA 196
SYGWDRIV macro 10
SYGWXPCH 10
Synchronous I/O indicator
property 183
System Application Architecture
discussion of 10
Systems Network Architecture
discussion of 11

T

TDPROC
Client-Library equivalent 41, 44
TDS
packet size property 184
TDS version property 184, 187
TDS version
symbolic values for 188
Text and image
text and image limit property 188
Text data
maximum length of text data property 184
Three-tier
discussion of 3, 4
Timeout
property 184
Timeouts
login timeout property 186
Transaction name
property 184, 188
Two-tier
discussion of 3, 7
Types. See Datatypes 173

U

User data
property 189
User name
property 189
User-allocated data
property 184
User-defined datatypes
definition 173
using in Client-Library programs 15

V

Variables
data descriptions in DATAFMT structure 170
Version
of Client-Library 159
TDS version property 184
Version number
Open ClientConnect property 189